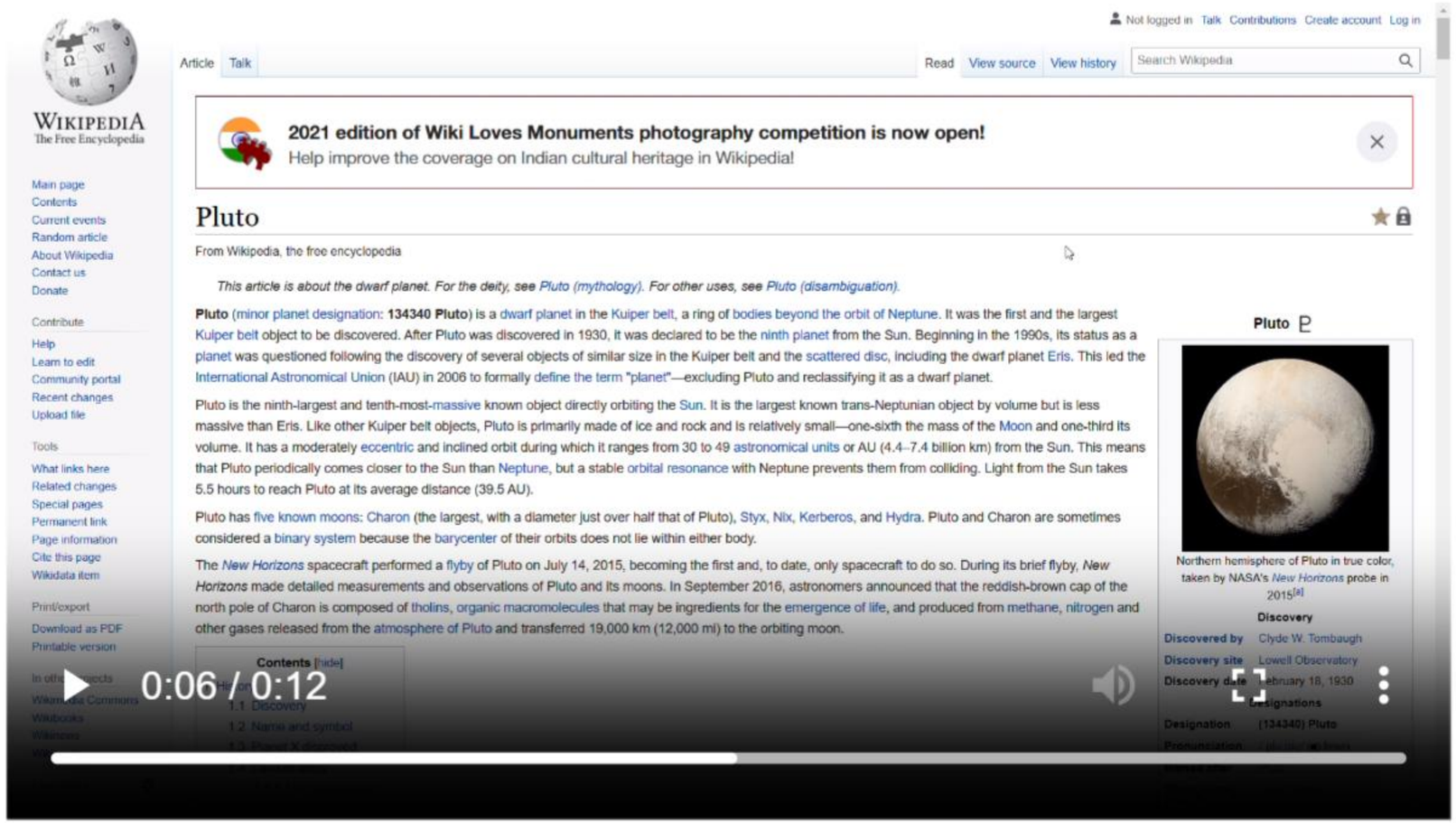
Slide Note



Desktop view of Wikipedia website

Do you think we should pinch-to-zoom to read the news content?

Menu

00:01:00:00:50 01/ 50

Slide Note

Mobile Optimized Websites

What do you see in the address bar of the browser showing an e-commerce website?

Slide Note

1. We can see that the URL in the above image has a normal URL, while in the below image the URL in the address bar is prefixed with the character **m**, meaning that it is optimized for mobile devices.

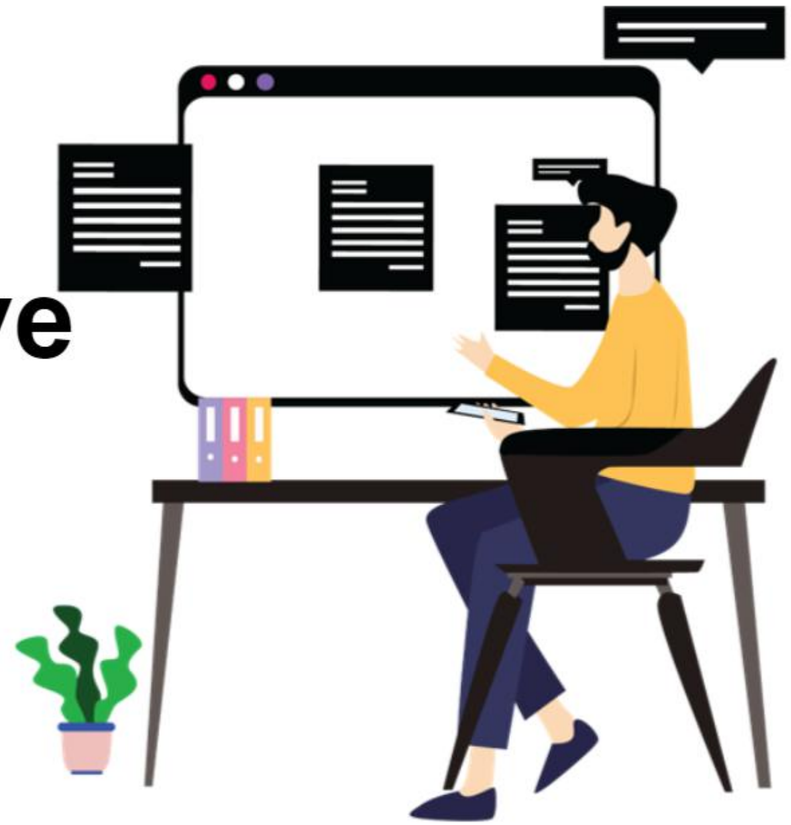
2. Because of the need to build two separate sites, one for desktops and one for mobile devices, additional costs are involved.

Mobile Optimized Websites

What do you see in the address bar of the browser showing an e-commerce website?

Slide Note

Develop a Responsive Web Page Using Modern CSS



Slide Note

Learning Objectives

- Explore responsive web design patterns
- Apply CSS display properties like flexbox and grid
- Create a responsive web design using flexible images
- Apply CSS media queries and breakpoints

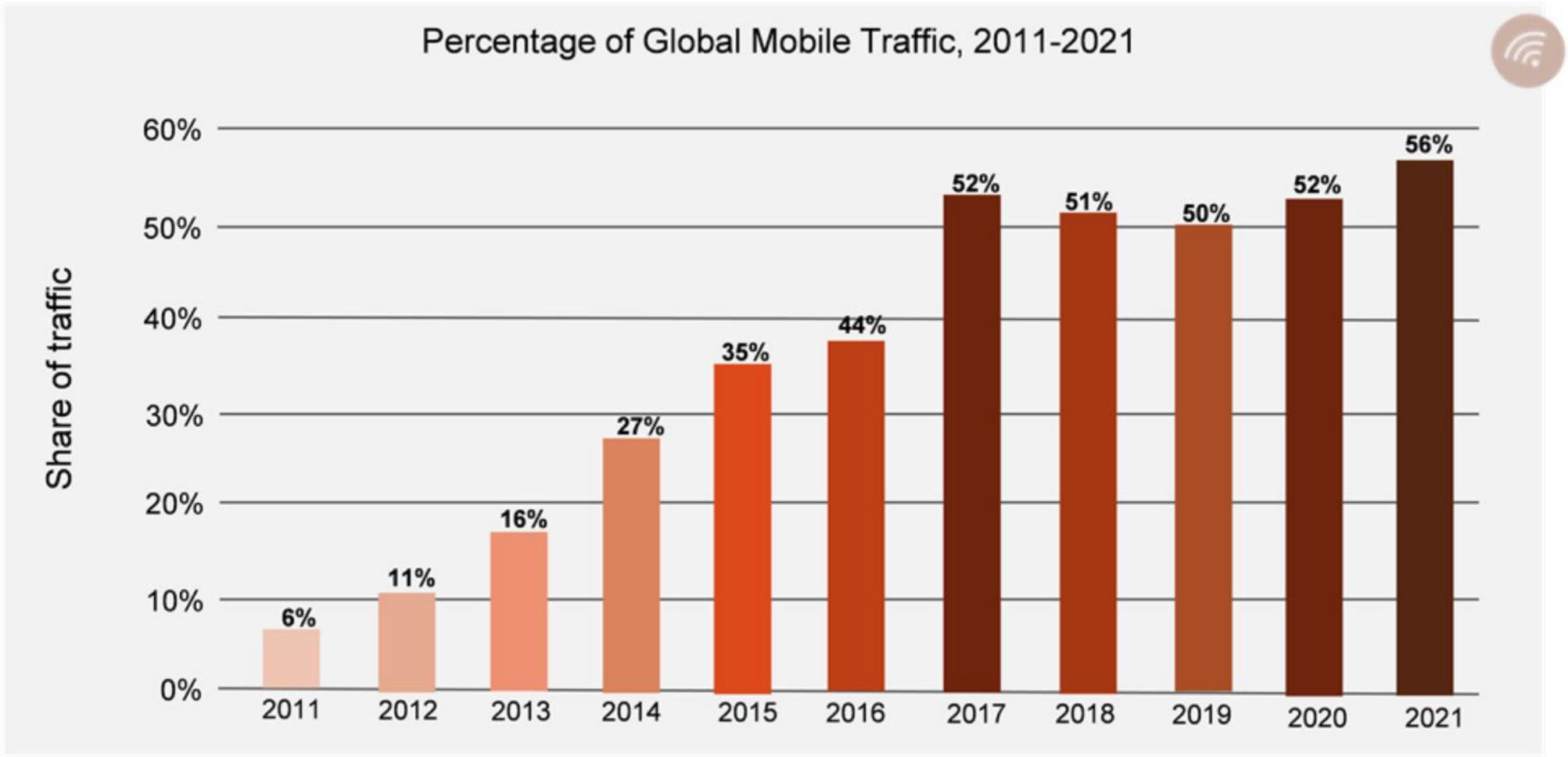
Slide Note

Mobile vs. Desktop usage today

As of January 2022, in terms of market share mobile has taken the lead at just above 55 percent of the market, with desktop devices taking up 42%.

Mobile websites have started becoming a lot more usable, sleek, and competitive. More people started getting smartphones and using them for a wide variety of uses.

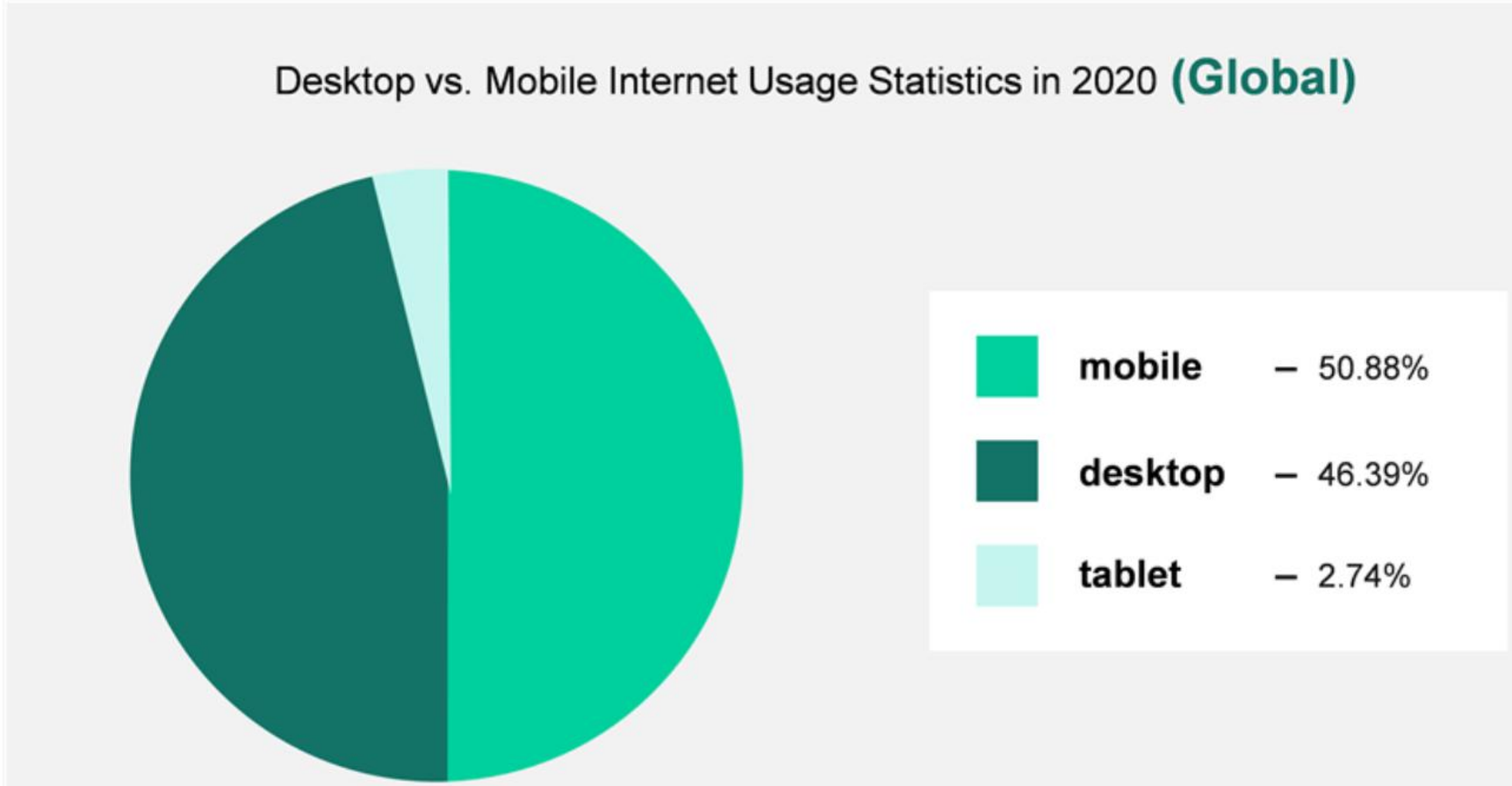
Percentage of Global Mobile Traffic



Slide Note

Today, the majority of internet traffic comes from mobile sources (and predominantly from smartphones). Globally, the latest statistics indicate that approximately 50.88% of all internet traffic can be attributed to mobile devices/smartphones.

Desktop vs. Mobile Internet Usage Statistics



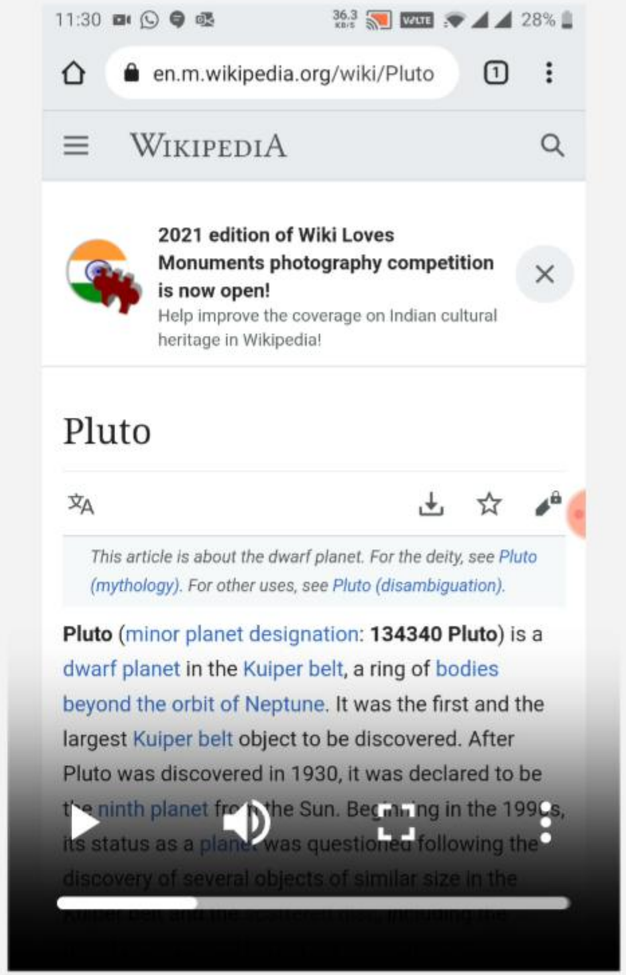
Slide Note

Think and Tell

- In the early days of web design, pages were designed for particular screen sizes.
- If the user's screen was larger or smaller than the expected design, results ranged from unwanted scrollbars to overly long line lengths, and poor use of space.
- As more diverse screen sizes are available, can we have a web design that allows web pages to alter their layout and appearance to suit different screen widths and resolutions?

Responsive Web Design

Slide Note



What Is Responsive Web Design?

A design that adapts itself to various screen sizes and orientations. It offers the following benefits:

- Smooth navigation
- Easy reading
- Minimum pinching
- Reducing scrolling and zooming
- Excellent user experience

Menu

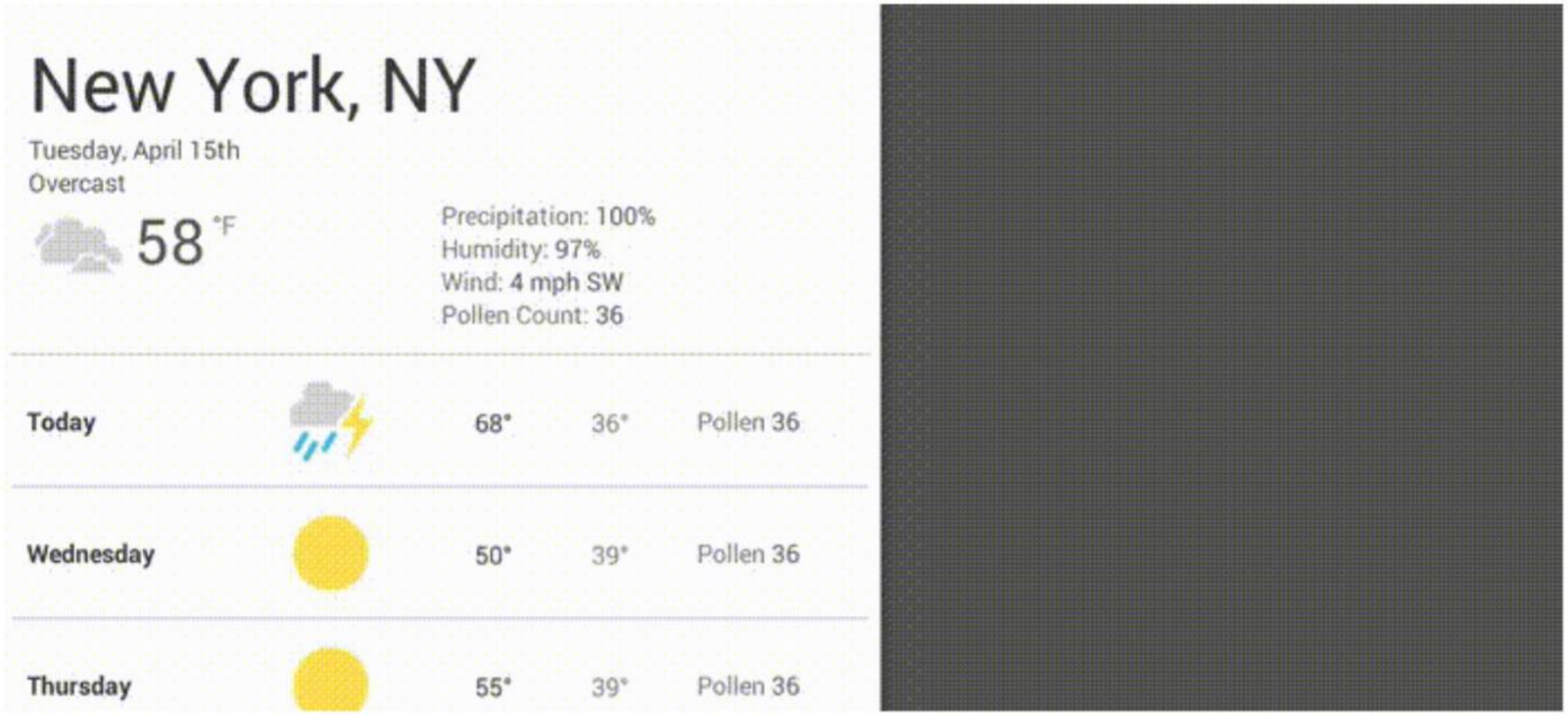
00:09:00:50

09/ 50

Slide Note

The layout changes based on the size and capabilities of the device. For example, on a phone users would see content shown in a single column view; a tablet might show the same content in two columns.

Responsive Web Design Example



Responsive web design responds to the needs of the users and the devices they are using.

Slide Note

- Web surfing using mobile devices has grown at an astronomical speed.

- Provides a seamless user experience.

- Cost-effectiveness: Developing a site for each device type involves cost. However, building a responsive site for each device type may not incur any additional cost.

- Easily adaptable to multiple device sizes.

- Easier to monitor analytics as there is only one website.

- Attracts more users and increases mobile traffic.

Benefits of a Responsive Web Design

Increased Web Surfing in Mobile Devices

Easy-to-Monitor Analytics

Seamless User Experience

Cost-effective

Easily Adaptable to Multiple Device Sizes

Attracts More Users

Slide Note

Consider a large website with many web pages. How can we fit various page elements within the viewport (visible area of the web page) without cluttering the page visually? How do designers fit links to many different site sections on a single page?

A drop-down menu will solve the challenge. This design pattern specifies a list of options when a mouse click or mouseover event occurs. The menu closes when the user hovers off it.

Web Design Pattern

- A set of guidelines for designing a user interface component is known as a web design pattern.
- These are developed to overcome specific user experience challenges and can be implemented by any website.
- It is a collection of best practices that helps to solve a user-centric problem.
- Web designers follow these patterns for the below reasons:
  - To ensure good UX.
  - Design patterns streamline the design process.

Slide Note

Responsive web design patterns are quickly evolving, but there are a handful of established patterns that work well across desktop and mobile devices.

Responsive Design Patterns

- Responsive design patterns are used to make a web page responsive across various devices of varying screen sizes.
- Most design patterns will use breakpoints to adapt to different screen sizes.
- A breakpoint is a point at which a different CSS will be applied, usually through a media query, to optimize the design for the user's viewport.
- The viewport is the user's visible area of a web page.
- The viewport varies with the device and will be small on a mobile phone than on a computer screen.
- A meta viewport tag gives the browser instructions on how to control the page dimensions and scaling.

Slide Note

Breakpoints for Various Devices

- **Breakpoints**: A breakpoint is the screen size threshold determined by specific layout requirements. At a given breakpoint range, the layout adjusts to suit the screen size and orientation.
- Some common breakpoints for devices with different screen sizes are:
  - 320px—480px: Mobile devices
  - 481px—768px: iPads, Tablets
  - 769px—1024px: Small screens, laptops
  - 1025px—1200px: Desktops, large screens
  - 1201px and more— Extra-large screens, TV

Slide Note

Most layouts used by responsive web pages can be categorized into one of the five patterns: mostly fluid, layout shifter, column drop, off canvas, and tiny tweaks.

In some cases, a page may use a combination of patterns, for example, column drop and off-canvas.

Some of the design patterns will be introduced in the session later.

Responsive Web Design Patterns

- Most layouts used by responsive web pages can be categorized into one of five patterns:
  - Mostly fluid
  - Layout shifter
  - Column drop
  - Off canvas
  - Tiny tweaks

These patterns, identified by Luke Wroblewski, provide a solid starting point for any responsive page.

Slide Note

Responsive design has several primary development components which are combined to create the entire design process.

Fluid Grid System: Fluid grids are based upon the sizing of page elements as opposed to using pixels. They're designed with proportions in mind so that the viewing screen will be properly fitted regardless of its size. They will also start using percentages for sizing.

Flexible images: The size of flexible images is dictated by relative units, so they are kept inside of their element. Adapt your images or other media to load differently depending on the device, either by scaling or by using the CSS overflow property.

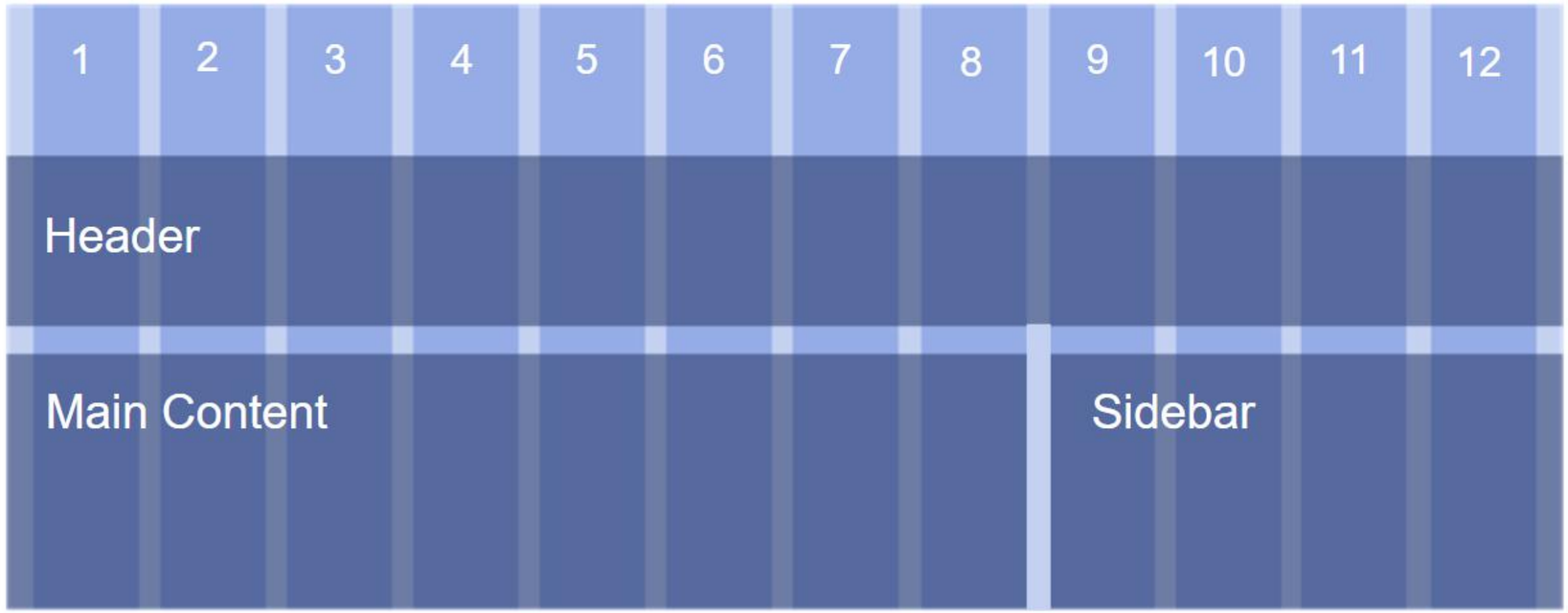
Media queries: Media queries allow the gathering of information about a site visitor. This information is then applied to CSS styles, based on the characteristics of the accessing device.

Main Components of a Responsive Web Design

Slide Note

Fluid Grid

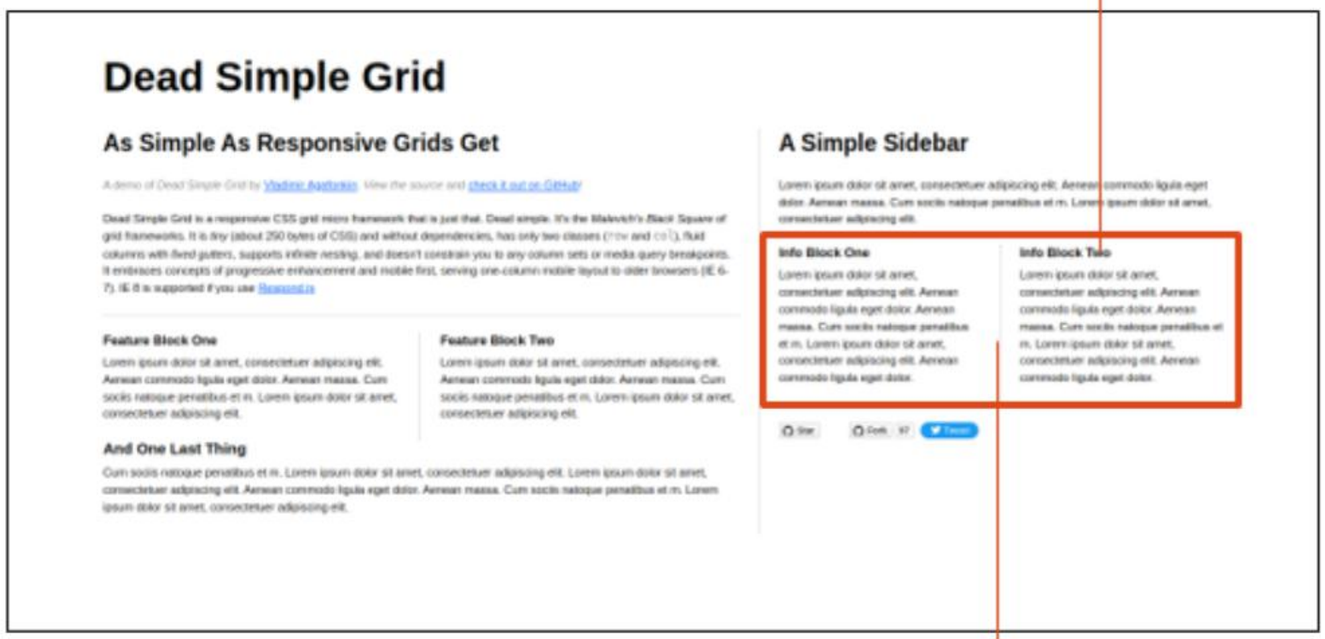
- A fluid grid used for responsive sites will ensure that the design is flexible and scalable.
- HTML elements will have consistent spacing and proportion and can adjust to a specific screen width based on percentages.



Slide Note

Desktop and Mobile View: Fluid Grid

Two Column layout



Gutter

Single Column Layout

Menu

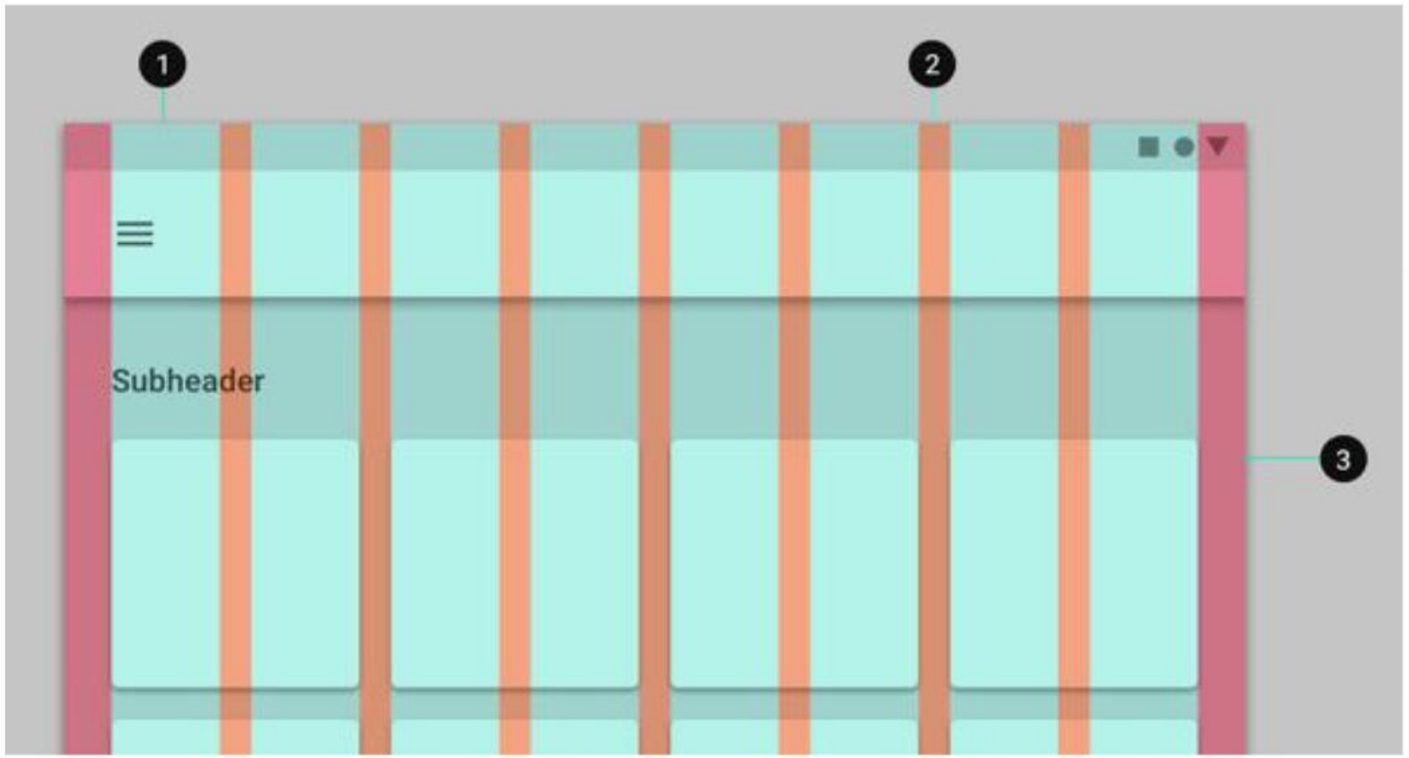
00:18:00:00:50

18/ 50

Slide Note

Responsive Grid System

The responsive layout grid is made up of three items: columns, gutters, and margins. The grid system can be implemented using **Floats**, **Flexbox**, or **Grid layout**.



Slide Note

Columns:

Content is placed in the areas of the screen that contain columns.

In responsive layouts, the column width is defined with percentages, rather than fixed values. This allows content to adapt to any screen size. The number of columns displayed in the grid is determined by the breakpoint range, a range of predetermined screen sizes. A breakpoint can correspond with a mobile, tablet, or any other screen type.

Gutters:

A gutter is a space between columns that helps separate the content.

Gutter widths are fixed values at each breakpoint range. To better adapt to a given screen size, gutter widths can change at different breakpoints.

Wider gutters are more appropriate for larger screens, as they create more open space between columns.

Margins:

Margins are the space between content and the left and right edges of the screen.

Margin widths are defined using fixed or scaling values at each breakpoint range. To better adapt to the screen, the margin width can change at different breakpoints. Wider margins are more appropriate for larger screens, as they create more whitespace around the perimeter of content.

Breakpoints:

A breakpoint is the screen size threshold determined by specific layout requirements. At a given breakpoint range, the layout adjusts to suit the screen size and orientation.

Each breakpoint range determines the number of columns, recommended margins, and gutters for each display size.

Responsive Grid System (cont'd)

- **Columns:** Content is placed in the areas of the screen that contain columns. In responsive layouts, the column width is defined with percentages, rather than fixed values.

- **Gutters:** A gutter is a space between columns that helps separate the content. Gutter widths are fixed values at each breakpoint range.

- **Margins:** Margins are the space between content and the left and right edges of the screen.

Slide Note

Flexbox Layout

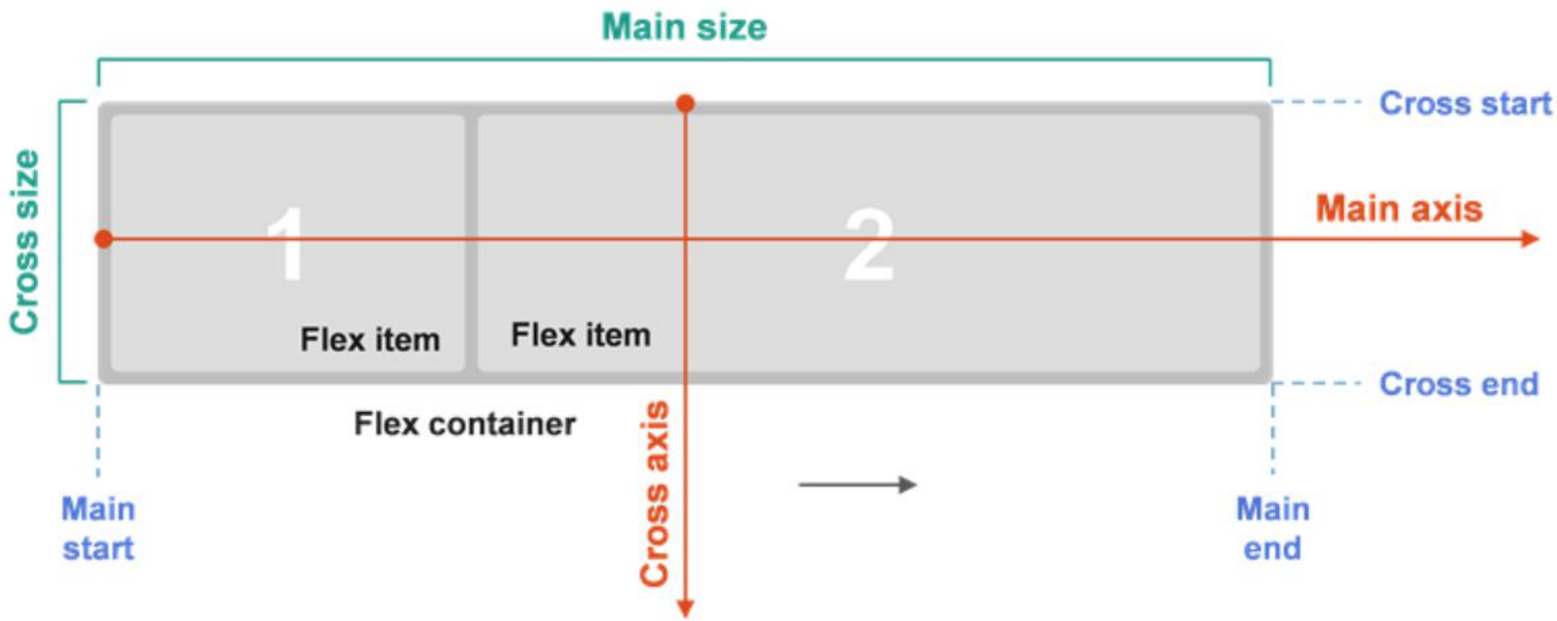
Slide Note

Flexbox Layout Is Simple and Powerful

- A flexible box layout is used for unknown or dynamic items.
- It's best for application components and small-scale layouts.
- Its structure contains a parent (flex container) and children (flex items).
- Flex items can "flex" their sizes to respond to their available spaces.
- It contains two axes, **main** and **cross,** which can be switched.

Slide Note

Flex Basics Terminology



Flex Container

```
div {
display: flex; /*or inline-flex*/
}
```

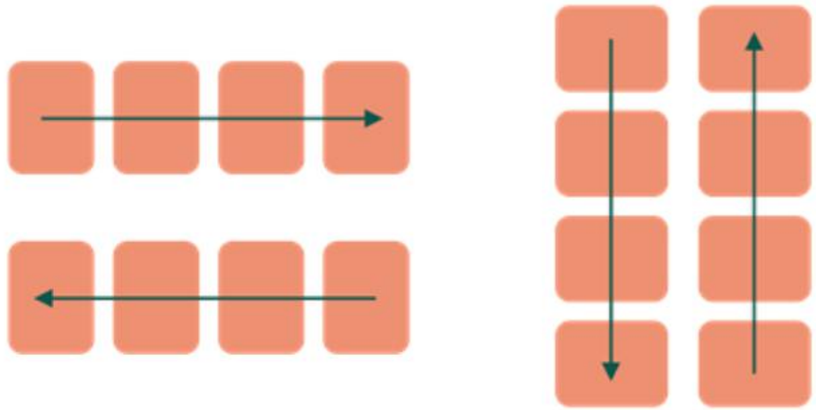
Slide Note

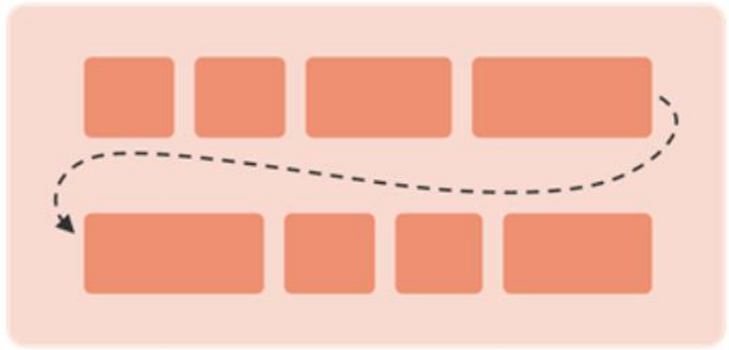
Flex-direction: This establishes the main axis and hence defines the direction for the flex items in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

Flex-wrap: By default, all flex items will try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

Flex-flow: This is shorthand for the flex-direction and flex-wrap properties, which together define the flex container's main and cross axes.

Flex Container (cont'd)





```
flex-direction: row | row-
reverse | column | column-
reverse ;
```

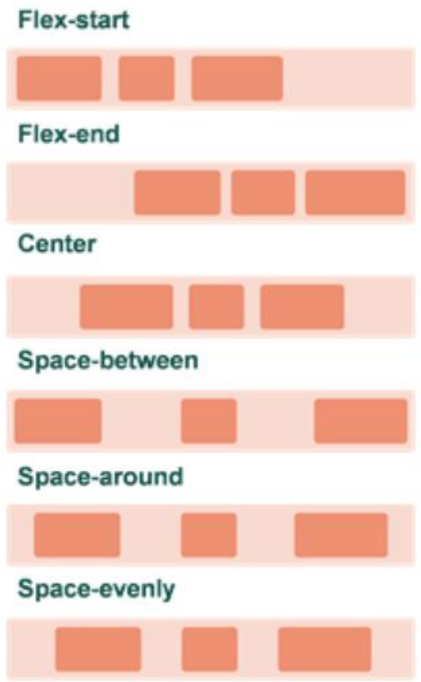
```
flex-wrap: wrap | nowrap |
wrap-reverse
flex-flow : row nowrap
/*default*/

flex-flow = flex-direction+
flex-wrap
```

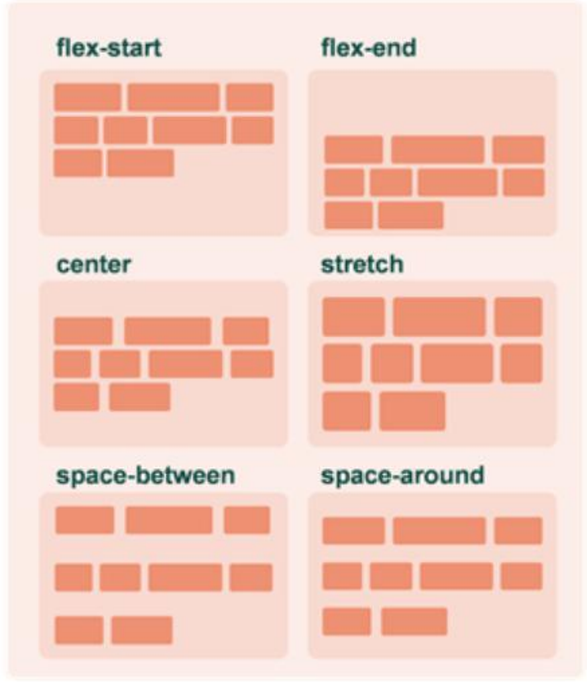
Slide Note

Justify-content: This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

Align-content: This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main axis.

Flex Container (cont'd)





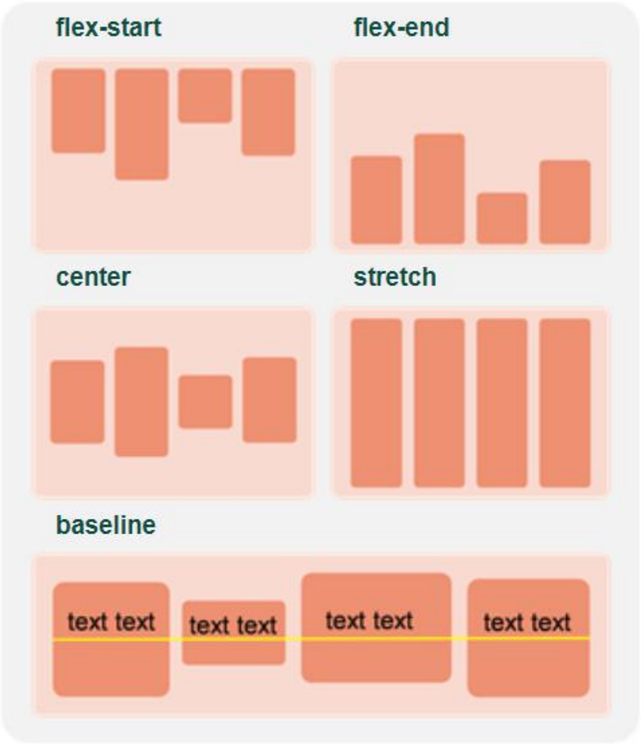

```
justify-content: flex-start
| flex-end | center |
space-between | space-
around;
```

```
align-content: flex-start |
flex-end | center | space-
between | space-around |
stretch;
```

Slide Note

Align-items: This defines the default behavior for how flex items are laid out along the **cross axis** on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main axis).

Flex Container (cont'd)




```
align-items: flex-start | flex-end |
center | baseline | stretch;
```

Slide Note

Use Chrome dev tools by inspecting the document, then use the toggle device icon to view various breakpoints depending on the screen width.

Use the drop-down menu next to the Responsive label to select from various screen widths.

Align Items Inside a Flex Container

Create flex containers and align items inside them using different values. Click on the following link to do so.

Flexbox Layout Demo

DEMO

Slide Note

Flex-grow: This defines the ability of a flex item to grow if necessary. If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, the remaining space would take up twice as much space as the others (or it will try to, at least).

In the left diagram top image, flex-grow:1 is implemented for all the items so that space is evenly distributed. In the bottom image, flex-grow:2 is implemented for the second item, taking twice as much as space as the others.

Flex-shrink: This defines the ability for a flex item to shrink if necessary.

Negative values are invalid for flex-grow and flex-shrink.

Flex-basis: This defines the default size of an element before the remaining space is distributed. If set to 0, the extra space around content isn't factored in. If set to auto, the extra space is distributed based on its flex-grow value.

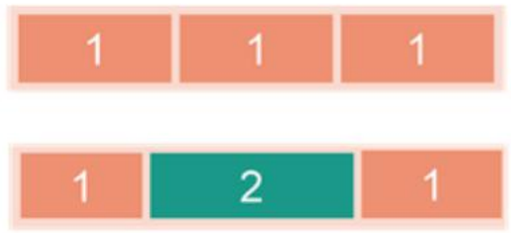
Flex shorthand property: This is shorthand for flex-grow, flex-shrink, and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional. The default is 0 1 auto, but if set with a single number value, it becomes 1 0.

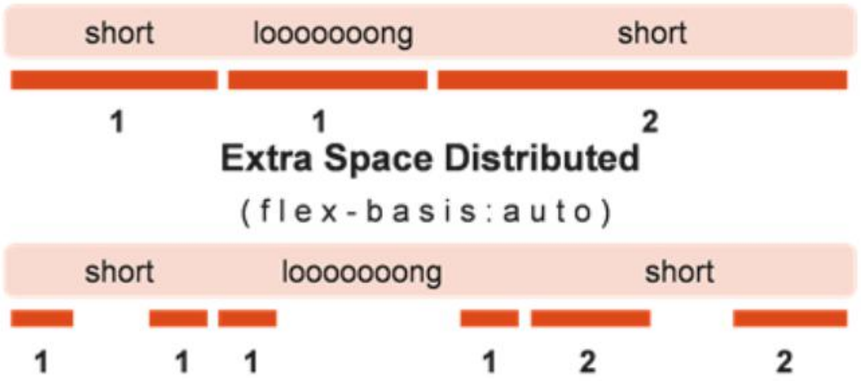
Note that float, clear, and vertical align have no effect on a flex item.

The right diagram shows the difference between absolute flex (starting from the basis of 0) and relative flex (starting from the basis of the item's content size). The three items have flex factors of '1', '1', '2' respectively. We can see that the item with the flex factor '2' grows twice as long as the others.

Flex Items

```
flex-grow : <'number'>
flex-shrink : </number'>
flex-basis: <'length'> | auto;
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```





Menu

00:28:00:00:50

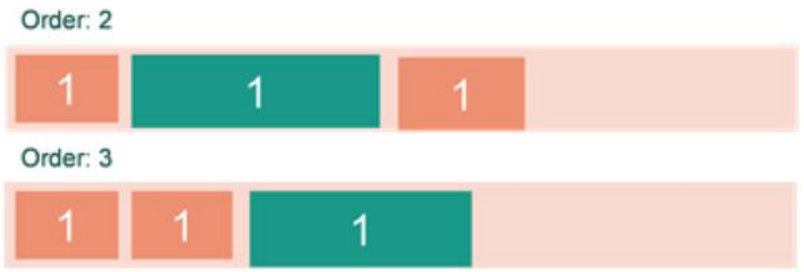
28/ 50

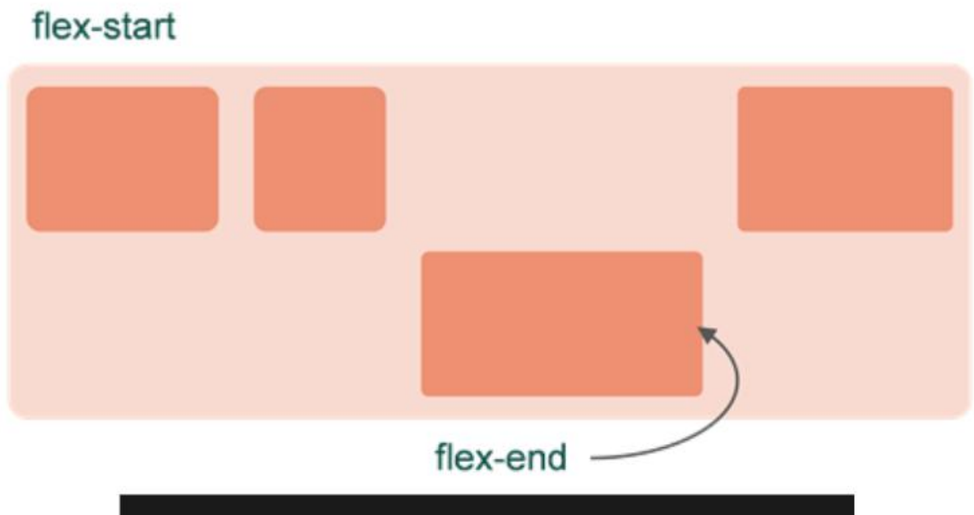
Slide Note

Align-self: This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.

Order: By default, flex items are laid out in the source order. However, the order property controls the order in which they appear in the flex container. Flex items with higher specified order values will appear later in the display order than items with lower order values.

Flex Items (cont'd)






```
order : <'integer'>
```

```
.item {
align-self: auto | flex-
start | flex-end | center |
baseline | stretch;
}
```

Slide Note

Quick Check

Which of the following choices is not a property for a flex container to align items?

a) display

b) flex-flow

c) flex-direction

d) justify-content

Slide Note

Quick Check: Solution

Which of the following choices is not a property for a flex container to align items?

a) **display**

b) flex-flow

c) flex-direction

d) justify-content

Explanation:

Options B, C, and D are used to align items inside the flex container

Slide Note

Quick Check

What is the default value for the flex-direction attribute?

a) row

b) row-reverse

c) column

d) column-reverse

Slide Note

Grid Layout

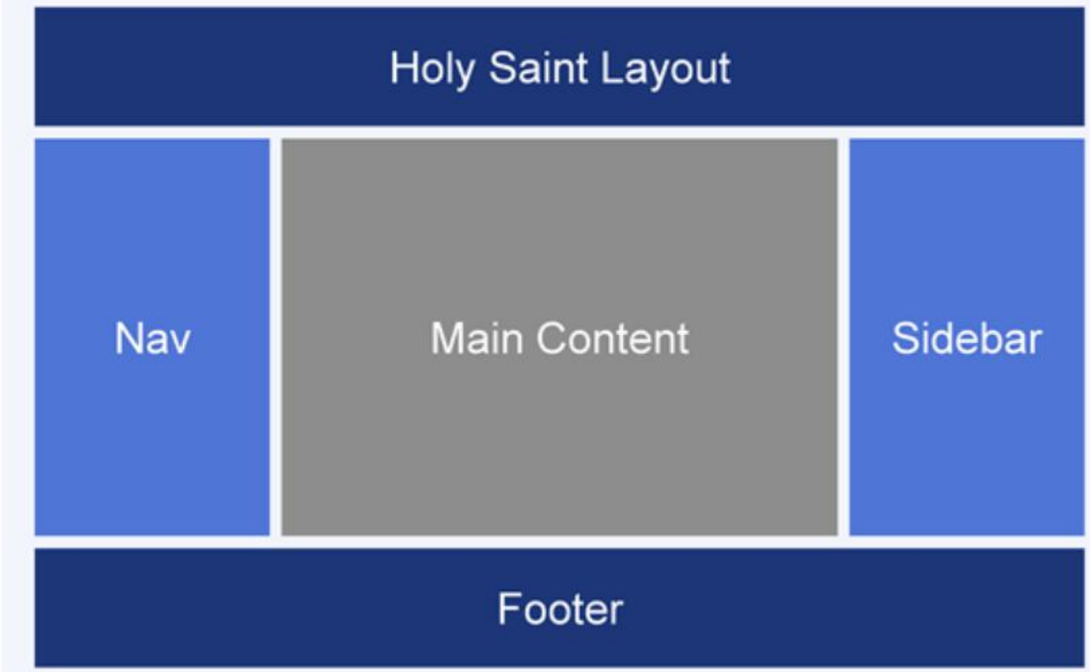
Slide Note

Grid Layout

- CSS module defines a two-dimensional grid-based layout system optimized for the user interface design.
- The children of a grid container can be positioned into arbitrary slots in a flexible or fixed predefined layout grid.
- A grid container establishes a new grid formatting context for its contents.
- Gridlines are either horizontal or vertical lines between grid cells.
- The grid cell is the smallest unit of the grid that can be referenced when positioning grid items.

Slide Note

CSS Grid Layout




```
Grid Container

div {
display: grid;
}
```

Slide Note

Grid Layout

Create a simple grid layout that has a header, a nav menu, the main article, an aside, and a footer. Let's take the help of the CSS Grid Layout link to do so.

DEMO

Slide Note

CSS Code:

#grid {
  display: grid;
  grid-template-rows: 1fr 1fr 1fr;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 2vw;
}
#grid > div {
  font-size: 5vw;
  padding: .5em;
  background: gold;
  text-align: center;
}

Here's an explanation of each declaration within that rule:

display: grid

Turns the element into a grid container. This is all that's required in order to create a grid. We now have a grid container and grid items.
The grid value generates a block-level grid container box. You can also use display: inline-grid to create an inline-level grid container box,
or display: subgrid to create a subgrid (this value is designed to be used on grid items themselves).

grid-template-rows: 1fr 1fr 1fr

Explicitly sets the rows of the grid. Each value represents the size of the row. In this case all values are 1fr (fractional unit), but they could have been done using a different unit, such as 100px, 7em, 30%, etc.

grid-template-columns: 1fr 1fr 1fr

Same as above except it defines the *columns* of the grid.

grid-gap: 2vw

Sets the gutter. The *gutter* is the gap in between the grid items. Here, we use the vw length unit (this is relative to the viewport's width), but we could just as easily have used 10px, 1em, etc.

The grid-gap property is actually a shorthand property for the grid-row-gap and grid-column-gap properties.

Basic Grid

Html Code:

```
<div id="grid">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    <div>9</div>
</div>
```

CSS Code :

```
#grid {
  display: grid;
  grid-template-rows: 1fr 1fr 1fr;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 2vw;
}

#grid > div {
  font-size: 5vw;
  padding: .5em;
  background: green;
  text-align: center;
}
```

Slide Note

Basic Grid (cont'd)

Output

1	2	3
4	5	5
6	7	8

Slide Note

max-width:

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size.

Picture Element:

The most common use of the element will be for images used in responsive designs. Instead of having one image that is scaled up or down based on the viewport width, multiple images can be designed to fill the browser viewport.

The srcset attribute is required, and defines the source of the image.

The media attribute is optional, and accepts the media queries you find in CSS @media rule.

You should also define an element for browsers that do not support the element.

with srcset and sizes attribute:

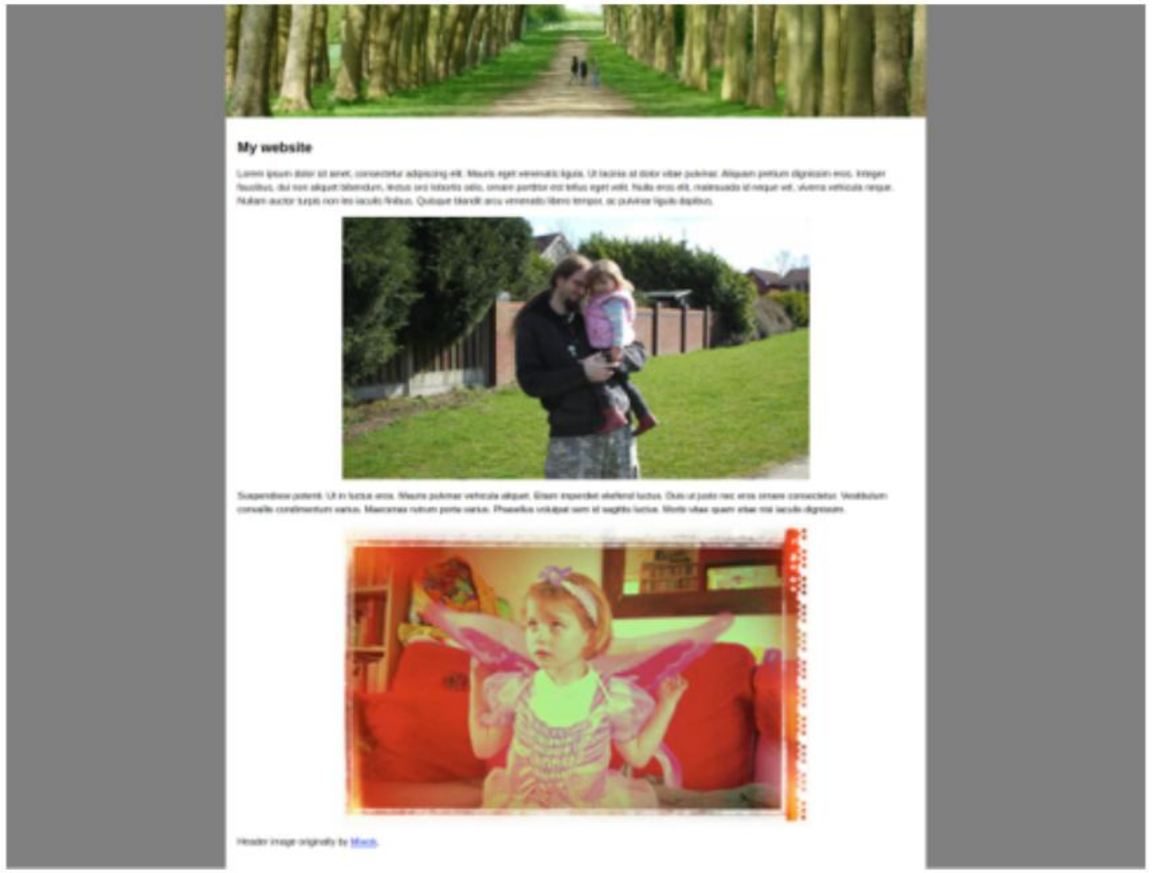
You can provide multiple sizes along with "hints" (meta data that describes the screen size and resolution the image is best suited for), and the browser will choose the most appropriate image for each device, ensuring that a user will download an image size appropriate for the device they are using.

Flexible Images

- The flexible images, also called adaptive images respond to different viewport sizes and display resolutions.
- Flexible images can be achieved by using the following:
  - max-width CSS property
  - <picture> element
  - <img> element with srcset and sizes attribute

Slide Note

Example: Flexible Images



Slide Note

Flexible Images

Create flexible images using different techniques. Let’s use the following link to look at these techniques.

Flexible Images Demo

DEMO

Menu

00:42 00:00:50

42/ 50

Slide Note

max-width:

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size.

Picture Element:

The most common use of the element will be for images used in responsive designs. Instead of having one image that is scaled up or down based on the viewport width, multiple images can be designed to fill the browser viewport.

The srcset attribute is required and defines the source of the image.

The media attribute is optional and accepts the media queries you find in the CSS @media rule.

You should also define an element for browsers that do not support the element.

Sample Code: Flexible Images

max-width Property

CSS Code

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

<img> Element

HTML Code

```
<img srcset ="elva-480w.jpg
480w, elva-800w.jpg 800w"
     sizes = "(max-
width:600px) 480px, 800px"
     src="elva-800w.jpg"
     alt="Elva fairy"
>
```

<picture> Element

HTML Code

```
<picture>
    <source srcset = "elva-
portrait.jpg" media="(max-
width: 799px)">
    <source srcset="elva-
800w.jpg" media="(min-
width: 800px)">
<img src="elva-
800w.jpg" alt="Chris with
Elva">
</picture>
```

Slide Note

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

@media not |
only *mediatype* and (*expressions*) {

CSS-Code;

}

The result of the query is "true" if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the "not" or "only" operators, the media type is optional, and the "all" type will be implied.

CSS Media Queries

- Media Query gives a way to apply CSS based on a device's general type (such as sprint vs. screen) or specific characteristics (such as screen resolution or browser viewport width).

- Simple media query syntax:

```
@media media-type and (media-feature-rule) {
        /* CSS rules go here */
         }
```

- It consists of:
  - A media type, which tells the browser what kind of media this code is for (Ex. print, or screen).
  - A media expression, which is a rule, or test that must be passed for the contained CSS to be applied.
  - A set of CSS rules that will be applied if the test passes and the media type is correct.

Slide Note

Sample Code: Media Queries

```
/*CSS style applied when device
width is greater than 480px*/
@media screen and (min-
width: 480px) {
   #leftsidebar {
      float: left;
      width: 200px;
   }
   #main {
      margin-left: 416px;
   }
}
```

```
/*CSS style applied when device
width is lesser than 480px*/
#leftsidebar {
   float: none;
   width: auto;
  }
#main {
   margin-left: 4px;
  }
```

Slide Note

Use this Github link to create a responsive grid layout.

Dead Simple Grid is a responsive CSS grid micro-framework/concept by Vladimir Agafonkin (creator of Leaflet) that is just that. Dead simple. It's *Malevich's Black Square* of grid frameworks.

tiny (about **250 bytes** of CSS), no dependencies

only **two classes** (row and col)

fluid columns with **fixed gutters**

supports **infinite nesting**

allows true **responsive design** (change column setup in media queries)

supports all major browsers starting from IE 8, serving one-column mobile layout to older browsers

built with progressive enhancement and mobile-first concepts in mind

CSS Media Queries

Create a responsive page content by adding a CSS media query for various screen widths. Let's look at the following link to see how this is done.
CSS Media Queries Demo Code

Design a responsive grid layout that adapts to the screen size. Click on Fluid Grid Demo Code to see how this is done.

DEMO

Slide Note

Responsive Web Page Using Media Queries

- Media queries can modify the appearance (and even behavior) of a website or an app by using the set of conditions about the user's device, browser, or system settings.
- They are a key part of responsive web design, as they allow us to create different layouts depending on the size of the viewport.
- It is a popular technique to deliver tailored stylesheets to desktops, laptops, tablets, and mobiles.
- These can be used to check the following:
  - Width and height of the viewport
  - Width and height of the device
  - Orientation (landscape or portrait) of the device
  - Resolution of the device

Slide Note

Quick Check

In CSS3, _____ is a rule used to apply a block of CSS properties.

a) @media
b) @property
c) @css
d) @media_query

Slide Note

Quick Check: Solution

In CSS3, _____ is a rule used to apply a block of CSS properties.

a) **@media**

b) @property

c) @css

d) @media_query

Menu

00:49/00:00:50

49/ 50