Module 3
CSS Grid

Session Objectives:

- Lay out an HTML5 page using CSS Grid
- Define a grid container using rows and columns
- Define named grid template areas
- Assign page elements to grid template areas for purposes of page layout
- Describe what Responsive Design is and what Mobile First is
- Create a page with multiple layouts depending on screen width
- Use media queries (screen width) to define different dimensions for grid containers
- Use the relevant dev tools available in Chrome or Firefox to assist with developing grid layouts

Module 3
CSS Grid

Layouts in CSS

Layouts for HTML pages used to be handled by the HTML side of web pages. At first, most of the layout was done using HTML tables and splitting the content up into different cells on the page.

Eventually, the layout was handled more by using <div>s and then trying to position those <div>s properly on the page using floats, alignments, and positioning. Then CSS libraries like Bootstrap came along to try and fix the layout issues in CSS. These became very popular and solved the issue as best they could.

But now there are better tools for handling layouts than ever before in CSS, and they are being supported by more and more browsers as they improve. The two main tools that you'll learn about are Flexbox and Grid. Both of these are rather new additions to CSS but are already being adopted by and added to all the major browsers.

Module 3
CSS Grid

CSS Grid Layout, or Grid

Grid is a fairly recent addition to CSS, but it's already adopted by more than 95% of browsers in use today. Grid provides the ability to split an HTML page into rows and columns and then assign elements of that page into the "grid" that is created.

Grid is typically used to split up the major sections of a page—like a header, footer, nav, and main content—and then assign elements to those sections. The browser then calculates and figures out how everything fits and how big it all should be.

Module 3
CSS Grid

Defining the Grid

HTML

```html
<body>
    <header>
    <h1>My Website!</h1>
    </header>
    <aside>Left sidebar</aside>
    <main>
    <h2>Main Content</h2>
    <p>This is the main site contents.</p>
    </main>
    <footer>copyright 2021, eula link, etc.</footer>
</body>
```

Module 3
CSS Grid

Defining the Grid

HTML


```
<body>
    <header>
    <h1>My Website!</h1>
    </header>
    <aside>Left sidebar</aside>
    <main>
    <h2>Main Content</h2>
    <p>This is the main site contents.</p>
    </main>
    <footer>Footer - copyright 2021, eula link, etc.</footer>
</body>
```
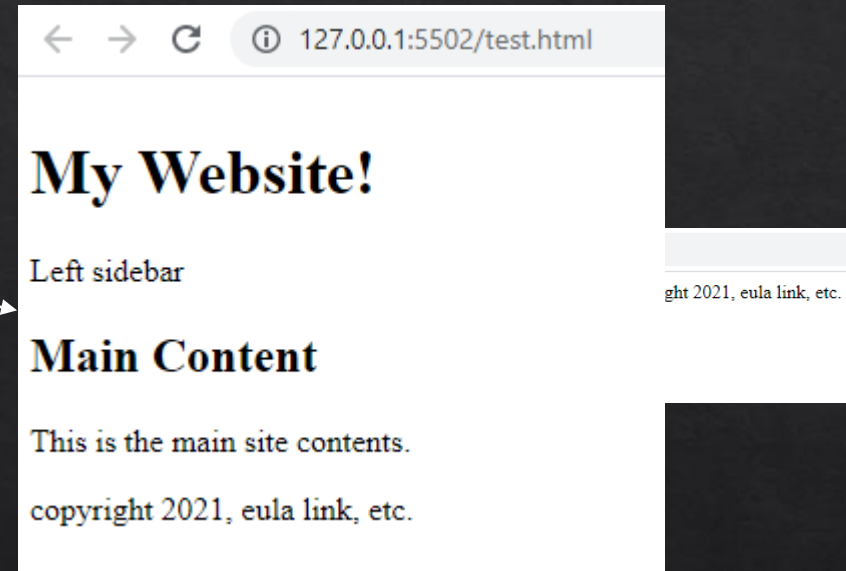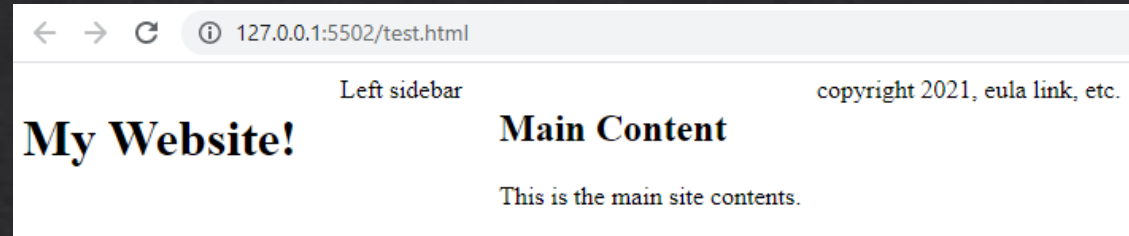
What if you have a requirement for the content to be laid out in four columns?

Module 3
CSS Grid

Defining the Grid

You can create this layout with grid. To define a grid, you'd use the grid value of the display property. Any element can be a grid container.

When you set display: grid, all of the direct children of the container become grid items:

CSS

```css
body {
    display:grid;
    …
}
```

The body is now a grid container, which by default gives you a one-column grid.

Module 3
CSS Grid

Grid template columns

To define the columns in your grid, you can use the property grid-template-columns. The value of this property is the number of columns you want to define and the size of the column.

CSS

```
body {
    display: grid;
    grid-template-columns: 200px 100px 200px 300px;  ⇦
}
```

Module 3
CSS Grid

Grid template columns

To define the columns in your grid, you can use the property grid-template-columns. The value of this property is the number of columns you want to define and the size of the column.

```css
body {
    display: grid;
    grid-template-columns: 1fr 1fr  1fr  1fr;
}
```
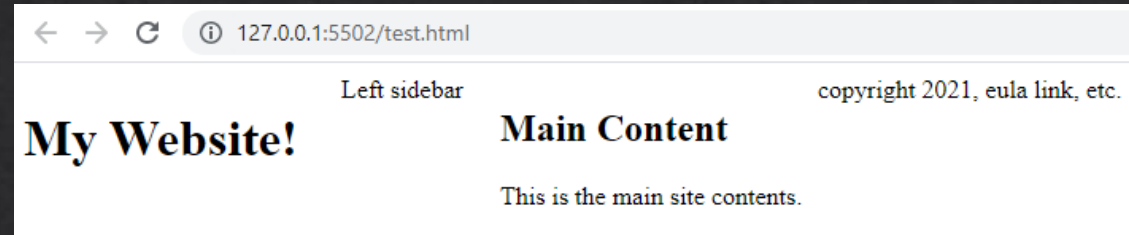
Grid Gap

Grid gap

You might also want some space between the grid items. You can do this by adding padding or margins to your elements, but it's better to add some space between the columns and rows instead. You can do this by adding the gap property to your grid definition:

CSS

```css
body {
    display: grid;
    grid-template-columns: 1fr 1fr  1fr  1fr;
    gap: 40px;

}
```

Grid template rows

To be more explicit with your row definitions in your grid, you can use
the property grid-template-rows.

HTML

```html
<div class="container">
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
    <div class="box">6</div>
    <div class="box">7</div>
    <div class="box">8</div>
    <div class="box">9</div>
    <div class="box">10</div>
    <div class="box">11</div>
    <div class="box">12</div>
</div>
```

CSS   Row display will be <u>implicitly</u> determined by grid

```css
.container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    gap: 20px;
}
```

Row display will be <u>explicitly</u> determined by using grid-template-rows

```css
.container {
    height: 500px;
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
⇨   grid-template-rows: 100px 1fr 100px;
    gap: 20px;
}
```

Module 3
CSS Grid

CSS Units: vh and vw

There are two CSS units of measure that allow you to stretch the entire height and width of the viewport

vh: Relative to 1% of the height of the viewport
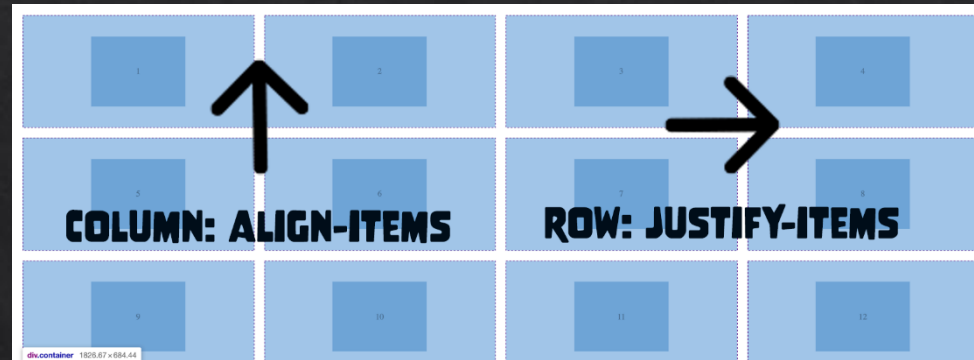vw: Relative to 1% of the width of the viewport

Module 3
CSS Grid

Box alignment in CSS Grid

When working with grid layouts, there are two axes you can align items against: the block axis (column) and the inline (row) axis.

These properties are defined on the grid container and define how all of the grid items are aligned:

- **align-items**: Aligns grid items along the block (column) axis

- **justify-items**: Aligns grid items along the inline (row) axis

*

Module 3
CSS Grid

Grid template areas

There are times when you'll need a grid item to span multiple columns, rows, or both. To do this, you use the grid property called grid-templates-areas.

With grid template areas, you define a grid of rows and columns that your page should be split up into, and then assign elements from your HTML into those grid areas. Using the same markup from before, you can create a grid that looks like this:
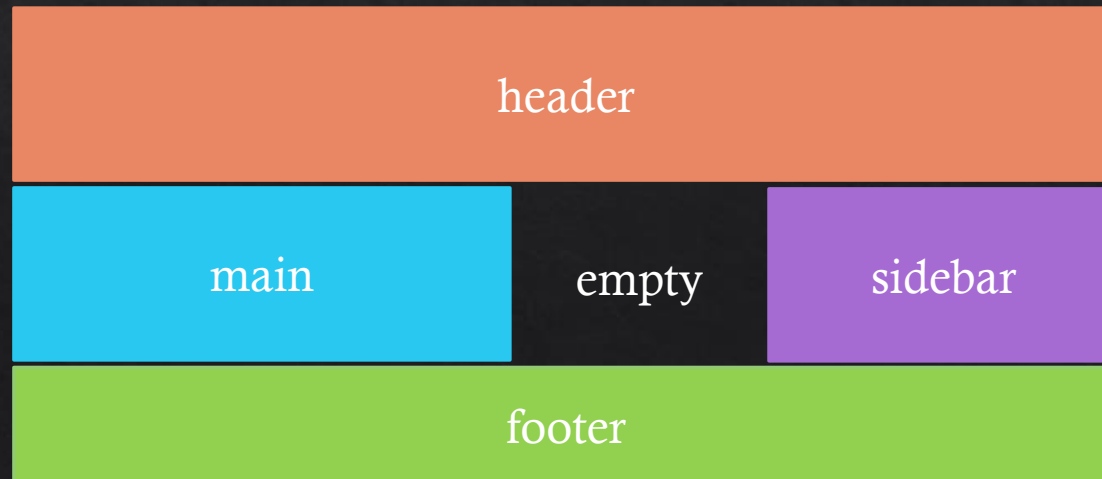
Grid template areas

You define the template in CSS, defining grid areas and naming them so that you can insert your content into them. First, you'd define the grid-template-areas property and set it to a string template.

If you define the same name to each column in a row, that means that element spans that entire row. If you put a . in one of the areas, that means that the area is empty.

You then have to give these grid names to your elements that are in the body element:

```css
body {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-template-areas:
        "header header header header"
        "content content . sidebar"
        "footer footer footer footer";
}

main {
    grid-area: content;
}

header {
    grid-area: header;
}

footer {
    grid-area: footer;
}

aside {
    grid-area: sidebar;
}
```

*

Module 3
CSS Grid

What's responsive web design?

Responsive web design is a set of practices that allows pages to alter their layout and appearance to suit different screen sizes and resolutions. The goal of responsive design is to create one site that works on every screen.

Designing a responsive interface
Applying a responsive design consists of addressing three key areas:

1. Flexible, or fluid, grid layouts
2. Resizable images
3. CSS media queries

Whether a visitor to your site uses a phone or a large screen desktop, your site must automatically switch to accommodate the screen's resolution and support larger image sizes. Even if the user doesn't switch devices, but changes the orientation of the screen from portrait to landscape, you want your design to accommodate the extra space and fill it with content accordingly.

*

Mobile first

With the shift in internet consumption coming from mobile devices, mobile screens are now the primary means in which users interact with your applications. As such, the term mobile-first was coined to indicate that applications should be developed with a "mobile-first mindset."

As developers, instead of adding breakpoints into the design as the width of the screen gets smaller, you should create breakpoints in the design when the width of the screen gets larger.

One approach that helps provide a mobile-first approach is to simulate mobile devices using Chrome or Firefox's Responsive Web Design tools.

A strategy to follow for introducing breakpoints is to start with the small screen first, then expand until it looks bad. At this point, it's time to insert a new breakpoint.