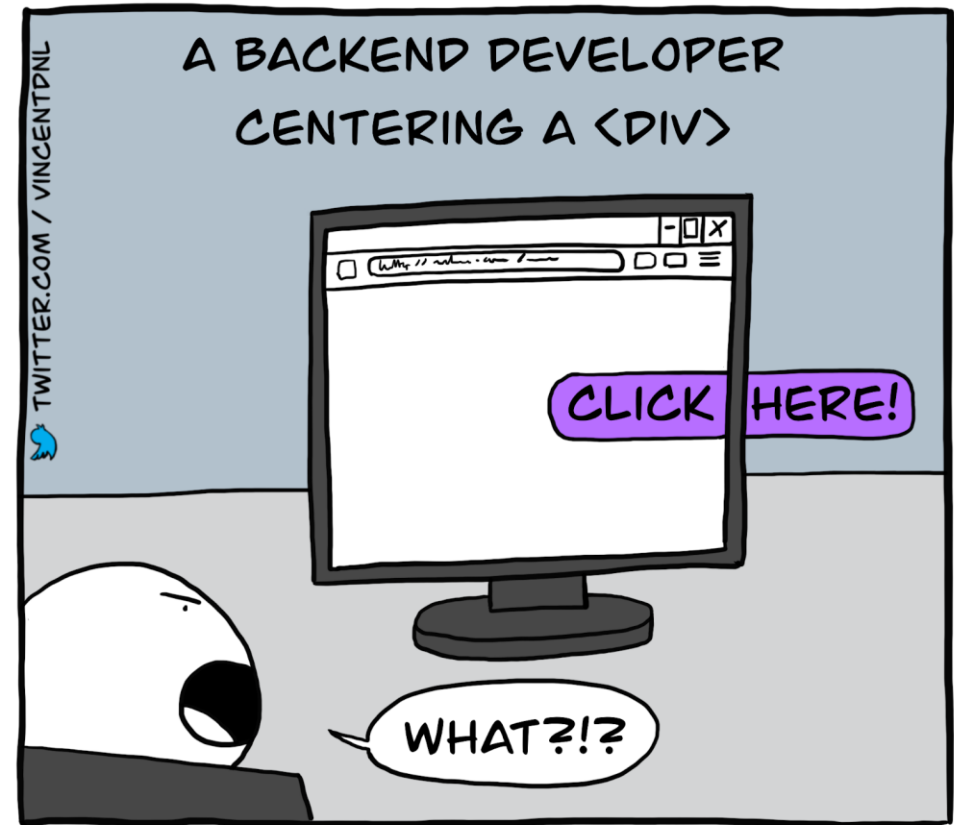


CSS JOKE

"Two CSS properties walk into a bar.
A barstool in a completely different
bar falls over."

Thomas Fuchs @thomasfuchs

```
.ninja {  
  color: black;  
  visibility: hidden;  
  animation-duration: 0.00001s;  
}
```

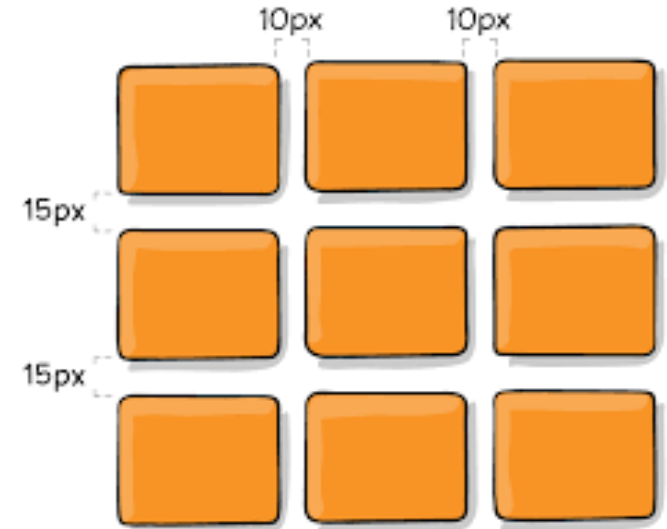


Module 3-3

CSS Grid and Responsive Design

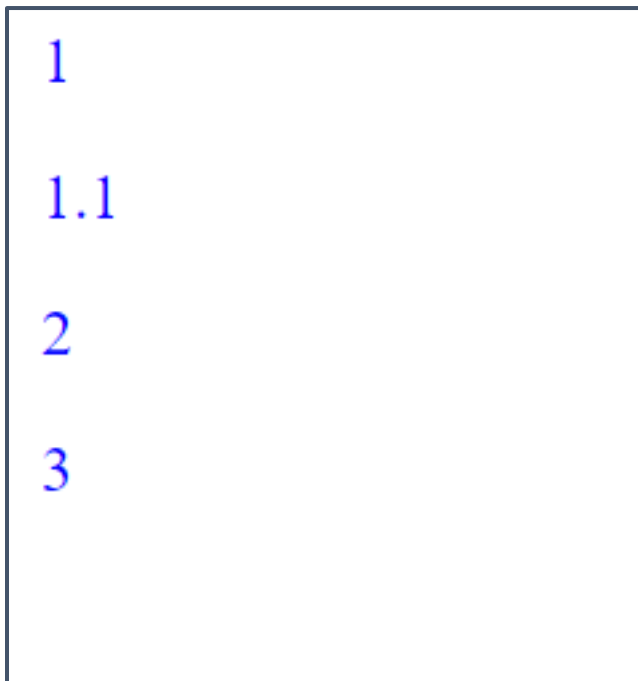
Objectives

- Learn about Grid container
- Define named grid template areas
- Assign page elements to grid template areas for page layout
- Describe what responsive design is and what mobile first is
- Use media queries to define different dimensions for grid containers
- Use browser developer tools to help developing grid layouts



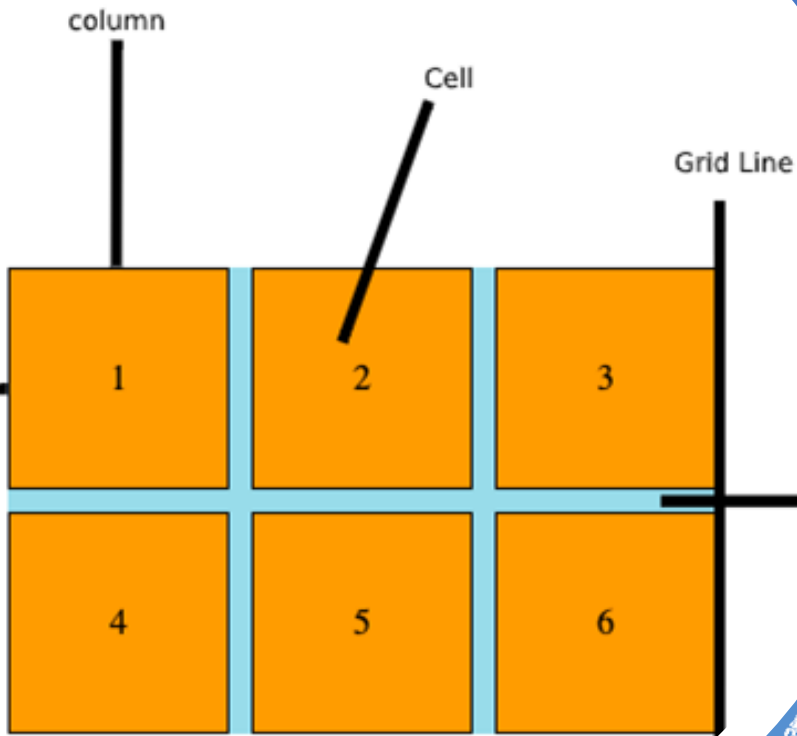
CSS Variables

```
:root {  
  --main-bg-color: blue;  
}  
  
div p {  
  color: var(--main-bg-color);  
}
```



CSS Grid: Introduction

- Grid allows us to write better layouts using in-browser capability of grids.
- Before grid, we had to either use something like Bootstrap or create a custom grid system.
- By defining a grid, we create a two-dimensional layout composed of columns and rows allowing us to better organize our web page's contents.



CSS Grids: Defining

To define a grid we must specify a display attribute with a value of grid:

```
.myGrid {  
    display: grid;  
}
```

In this example, the CSS code will specify using a selector by class, that an html element with a class name of myGrid be defined as a grid. All of the direct children of the container will become grid items.

CSS Grids: Columns and grid-gap

grid-template-columns: This property defines the number of columns (and their respective width).

grid-gap: Creates a gutter, setting a width of space between the columns and rows. (note – not around outside of container)

```
body {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr 2fr 2fr 1fr;  
  gap: 40px;  
}
```

fr stands for fractional unit. The 1st column will occupy 10% of the width, the second 20%, etc.


Adds a buffer between the cells of the grid.

CSS Grids: Template Areas

grid-template-areas: Matches each area of the grid to a specific HTML element. By virtue of how this works, it also defines the number of rows.

```
body {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr 2fr 2fr 1fr;  
  gap: 20px;  
  grid-template-areas:  
    ". header header nav  nav  ."  
    ". main  main  main main  ."  
    ". fall-festival fall-festival store store .";  
}
```

The period represents an empty space in the final grid layout.



CSS Grids: Template Areas

grid-areas: Allows you to name each element specifically to be able to access it, or allows to give a name for each element to be displayed in the grid.

```
header {  
  grid-area: header;  
}  
  
nav#menu-nav {  
  grid-area: menu-nav;  
}  
  
main {  
  grid-area: main;  
}  
  
aside {  
  grid-area: upcoming-events;  
}  
  
footer {  
  grid-area: footer;  
}  
  
.container {  
  display: grid;  
  grid-template-columns: 200px 1fr 200px;  
  grid-template-areas:  
    "header header header"  
    "menu-nav main upcoming-events"  
    "footer footer footer"  
  ;  
  height: 100vh;  
  grid-gap: 10px;  
}
```

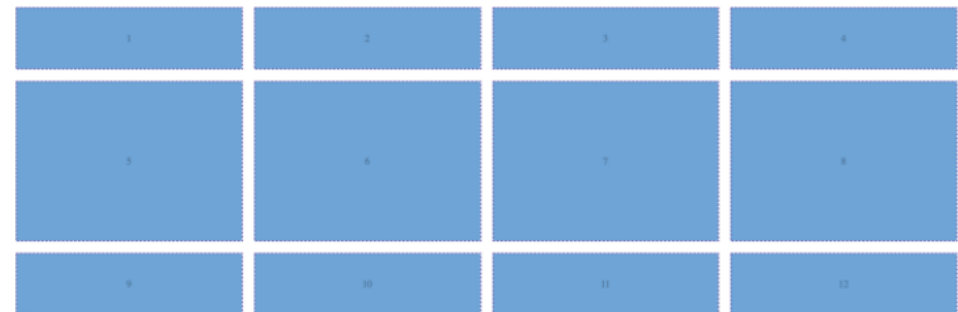
grid-template-area
property matches up to
the grid-area property

CSS Grids: Template Rows

grid-template-rows: Specifically allow you to specify how many rows.

```
body {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr 1fr;  
  grid-template-rows: 100px 1fr 100px;  
  grid-template-areas:  
    "header header nav nav "  
    "main main main main"  
    "fall-festival fall-festival store store";  
}
```

First and third row have a set height of 100px.
Second row will take up the rest of the available space.



CSS Units: viewport vs. pixels vs. fractional units

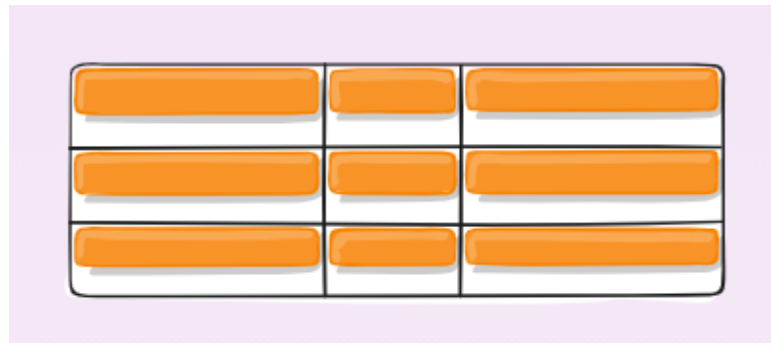
- Pixels are fixed length units. 100px is the same size no matter how big the display (viewport)
- vh (viewport height) and vw (viewport width) can be used to set the height and/or width the container (1vh is 1% of viewport height)
- fr (fractional units) typically used to specify how much of the container each grid item is allotted

```
body {  
  margin: 0;  
  padding: 0;  
}  
  
.container {  
  height: 100vh;  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  grid-gap: 20px;  
}
```

CSS Grids: Align-items

Aligns grid items along the block (column) axis. Used at the container level.

```
body {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr 2fr 2fr 1fr;  
  grid-template-rows: 100px 1fr 100px;  
  align-items: start;  
}
```



Start – flush with start edge of cell

End – flush with end edge of cell

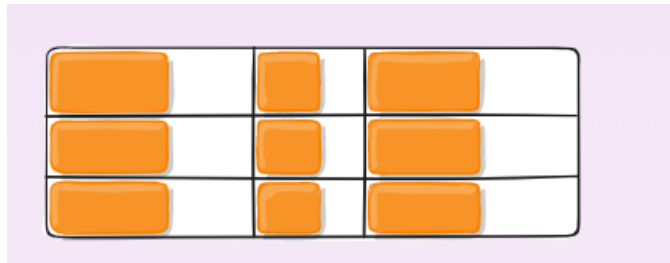
Center – centers in cell

Stretch – fills whole height of cell (default)

CSS Grids: Justify-items

Aligns grid items along the inline (row) axis. Used at the container level.

```
body {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr 2fr 2fr 1fr;  
  grid-template-rows: 100px 1fr 100px;  
  justify-items: start;  
}
```



Start – flush with start edge of cell

End – flush with end edge of cell

Center – centers in cell

Stretch – fills whole width of cell (default)

Let's do some coding!

RESPONSIVE DESIGN

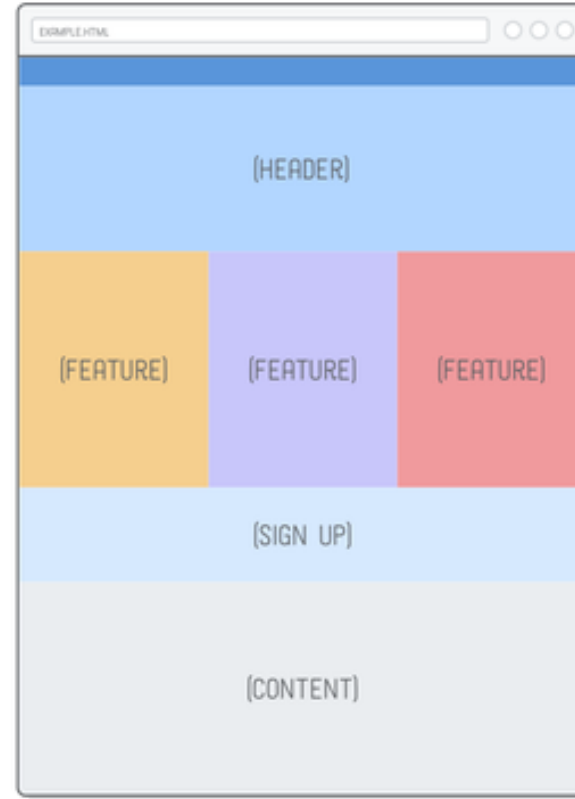
MOBILE



TABLET



DESKTOP



RWD: Setting the viewport

- Add meta tag to all web pages:

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

- This sets viewport of the page, giving browser instructions on how to control dimension and scaling

RWD: creating responsive images

- Allowing images to scale to fit any browser size.

```

```

- Above will allow image to be scaled larger than original size. Can set max-width instead – image will scale down but never larger than original size

```

```

RWD: creating responsive text

- Allowing text to scale to fit any browser size.

```
<h1 style="font-size:10vw">Hello World</h1>
```

- Depending on size of viewport, text will grow or shrink to scale.

Responsive Text

Resize the browser window to see how the text size scales.



RESPONSIVE DESIGN

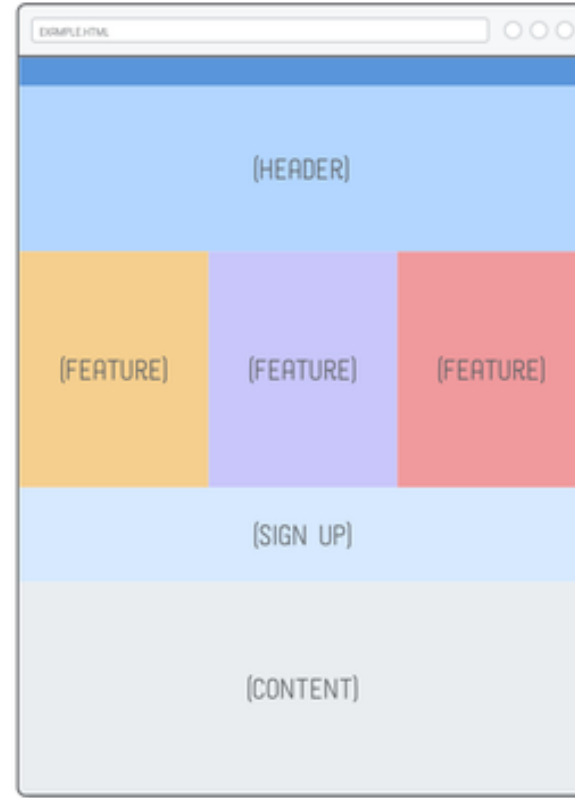
MOBILE



TABLET

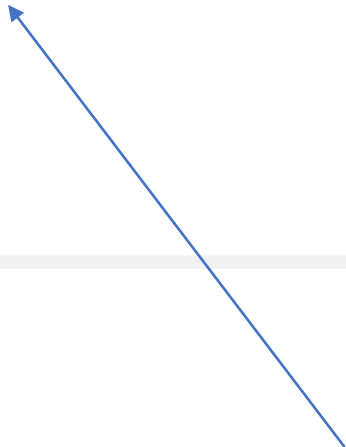


DESKTOP



MEDIA QUERIES

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

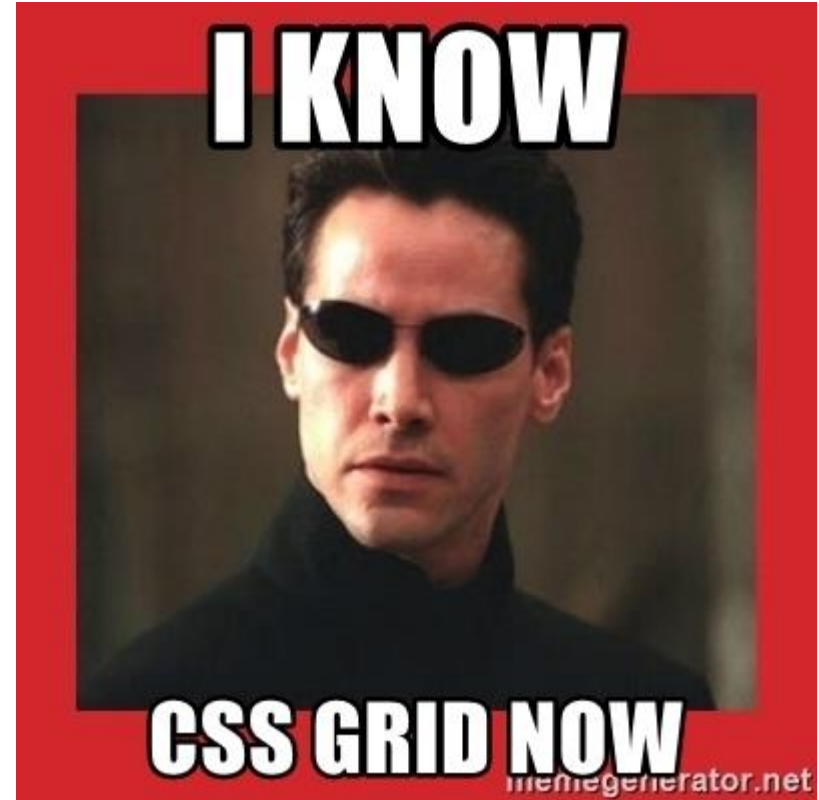


For screens up to the maximum width of 600px, the background color will be lightblue.

Let's do some coding!

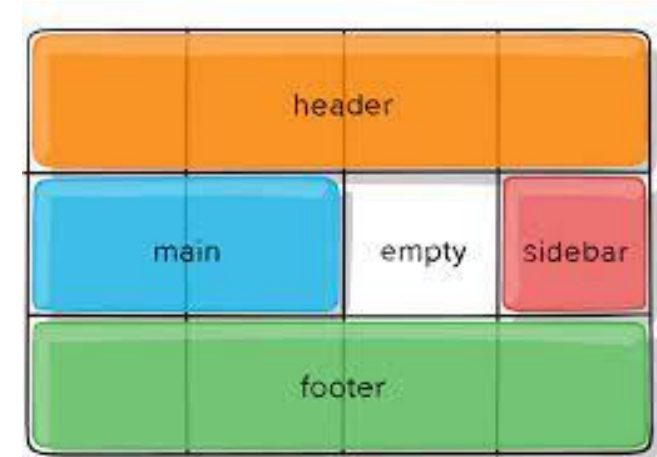
Objectives

- Learn about Grid container



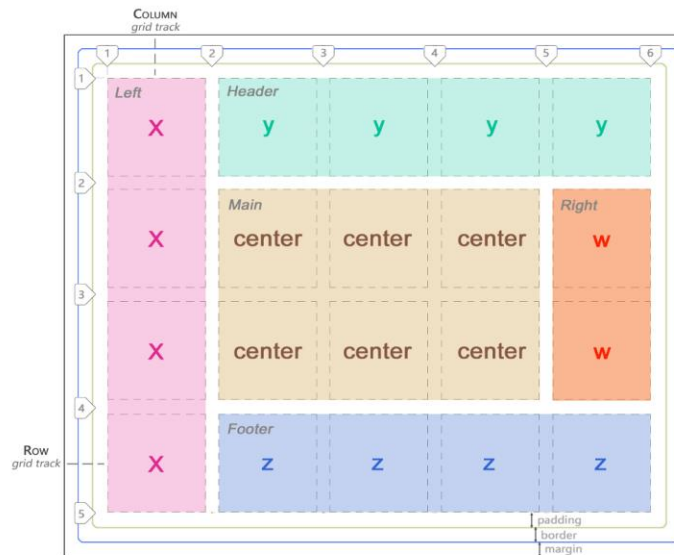
Objectives

- Learn about Grid container
- Define named grid template areas



Objectives

- Learn about Grid container
- Define named grid template areas
- Assign page elements to grid template areas for page layout



```
1 <head>
2   <style type = "text/css">
3     #grid { grid-template-areas:
4       'x y y y y';
5       'x center center center w';
6       'x center center center w';
7       'x z z z z';
8     }
9   </style>
10 </head>
11 <body>
12   <div id = "grid">
13     <div style = "grid-area: x">Left</div>
14     <div style = "grid-area: y">Header</div>
15     <div style = "grid-area: z">Footer</div>
16     <div style = "grid-area: w">Right</div>
17     <div style = "grid-area: center">Main</div>
18   </div>
19 </body>
20
```


Objectives

- Learn about Grid container
- Define named grid template areas
- Assign page elements to grid template areas for page layout
- Describe what responsive design is and what mobile first is



Objectives

- Learn about Grid container
- Define named grid template areas
- Assign page elements to grid template areas for page layout
- Describe what responsive design is and what mobile first is
- User media queries to define different dimensions for grid containers

```
/* normal style */
#header-image {
    background-repeat: no-repeat;
    background-image: url('image.gif');
}

/* show a larger image when you're on a big screen */
@media screen and (min-width: 1200px) {
    #header-image {
        background-image: url('large-image.gif');
    }
}

/* remove header image when printing. */
@media print {
    #header-image {
        display: none;
    }
}
```



Objectives

- Learn about Grid container
- Define named grid template areas
- Assign page elements to grid template areas for page layout
- Describe what responsive design is and what mobile first is
- User media queries to define different dimensions for grid containers
- Use browser developer tools to help developing grid layouts

