React – Flexbox

In this lab, you will create a Kraken app that uses the CSS style, flexbox.

# Objectives

In this lab, you will

* Create a Kraken application
* Create an HTML page
* Using flexbox, create a page with a header, footer, main, and nav bars
* Run the app

# Create the Kraken App

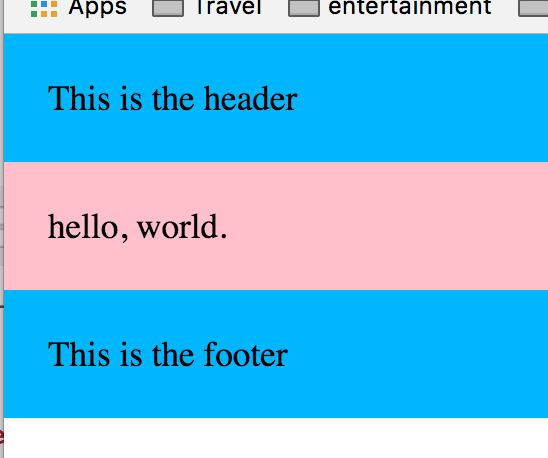
1. Create a Kraken app named Flexbox and accept all the defaults



1. Create a file, flexbox.html , in the /public folder and add the following text:

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Flexbox</**title**>  
 <**style**>  
 **html**, **body** {  
 **height**:100%;  
 **padding**: 0**px**;  
 **margin**: 0**px**;  
 }  
 **body** {  
 }  
 **header** {  
 **background-color**: **#00B7FF**;  
 **padding**: 20**px**;  
 }  
 **footer** {  
 **background-color**: **#00B7FF**;  
 **padding**: 20**px**;  
 }  
 **main** {  
 **background-color**: **pink**;  
 **padding**: 20**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**header**>  
 This is the header  
</**header**>  
<**main**>  
  
 hello, world.  
  
</**main**>  
  
<**footer**>  
 This is the footer  
</**footer**>  
</**body**>  
</**html**>

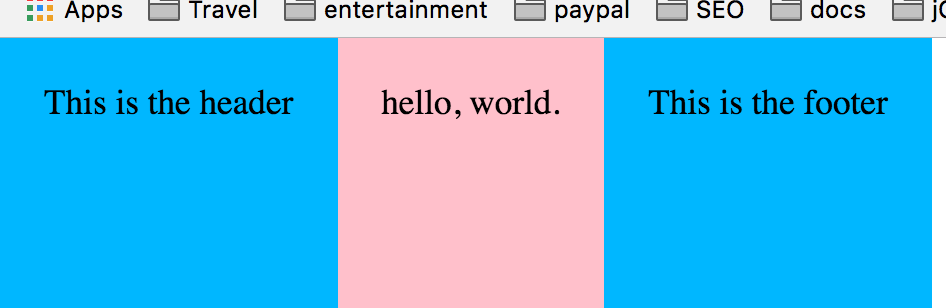
1. Test the page in the browser with: <http://localhost:8000/flexbox.html>



1. We will use CSS to change the styling of the page.
2. NOTE: the <html> and <body> tags have height set to 100%. We will use this later to put the <footer> at the bottom. If the height of the <body> were not defined, it would shrink to fit the contents.
3. NOTE: The <body> tag is the parent of the <header>, <footer>, and <main>. We will change the style of the <body> as the flex container and change the others as flex children
4. Currently, the display property of all components is block. Let’s change the display property of <body> to be flex. Edit the file and add the yellow line:

<**style**>  
 **html**, **body** {  
 **height**:100%;  
 **padding**: 0**px**;  
 **margin**: 0**px**;  
 }  
 **body** {  
 **display**: **flex**;  
 }  
 **header** {

1. Refresh the page and notice that the <body> layout is now flex with default direction being row. The page should look like:



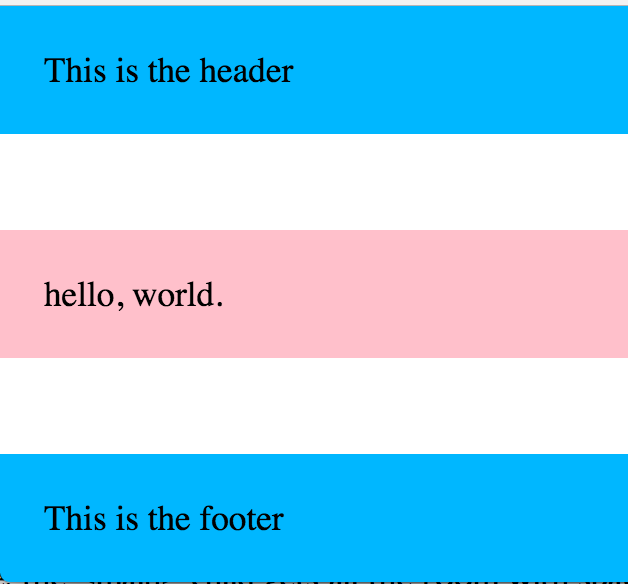
1. Note that the children of <body> appear inline-block instead of block.
2. Change the flex-direction property for the <body> tag to column and refresh the page.

**body** {  
 **display**: **flex**;  
 **flex-direction**: **column**;  
}

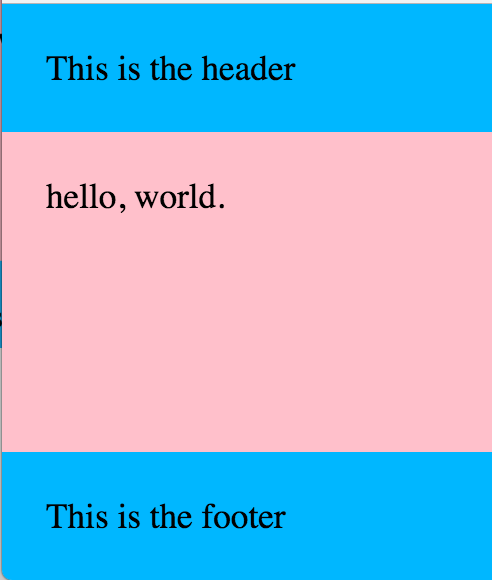
1. We are back where we started!!!
2. Let’s put some spacing between the three tags in <body>. Let the <header> be at the top of the page, <footer> be at the bottom and <main> be in the middle.

**body** {  
 **display**: **flex**;  
 **flex-direction**: **column**;  
 **justify-content**: **space-between**;  
}

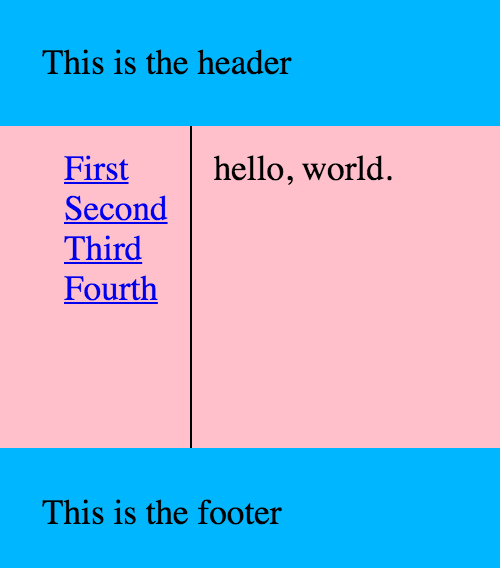
1. The justify-content value of space-between will put the first child at the top and the last child at the bottom and split the spacing evenly for all other children. In this case, the <main> child gets all the room with spacing above and below it. It should appear like:



1. Let’s change the height of the <main> tag to be 100% also. Now, the <main> tag takes all the space between the <header> and the <footer> tags. As the page height increases or decreases, all the changes occur to the <main> tag.



1. Let’s build the following page next:



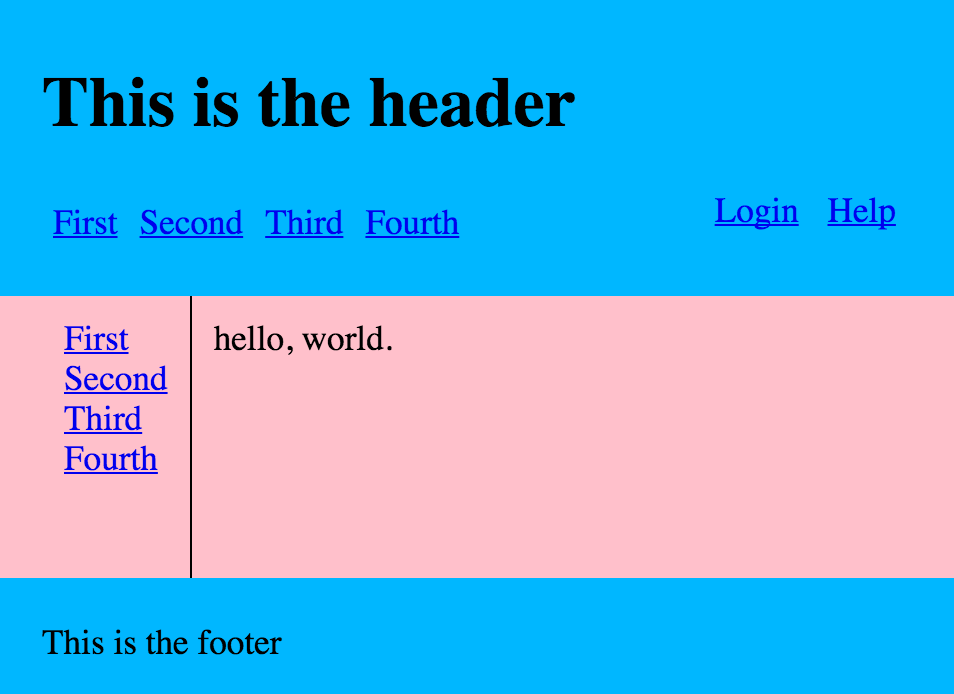
1. The HTML looks like the following:

<**header**>  
 This is the header  
</**header**>  
<**main**>  
  
 <**nav**>  
 <**a href="#"**>First</**a**>  
 <**a href="#"**>Second</**a**>  
 <**a href="#"**>Third</**a**>  
 <**a href="#"**>Fourth</**a**>  
 </**nav**>  
 <**div class="mainBody"**>  
 hello, world.  
 </**div**>  
</**main**>  
  
<**footer**>  
 This is the footer  
</**footer**>

1. Notice that <main> now has two children, <nav> and <div>.
   1. Using flex, is the flex direction of <main> a row or a column?
   2. Using flex, is the flex direction of <nav> a row or a column?
2. Let’s set up the CSS for the page now. Change the red content to the string, ‘row’ or ‘column’.

**main** {  
 **background-color**: **pink**;  
 **padding-left**: 20**px**;  
 **height**: 100%;  
 **display**: **flex**;  
 **flex-direction**: **???**;  
}  
**main nav** {  
 **height**: 100%;  
 **padding**: 10**px**;  
 **display**: **flex**;  
 **flex-direction**: **???**;  
 **border-right**: 1**px solid black**;  
}  
**div**.**mainBody** {  
 **padding**: 10**px**;  
}

1. Let’s add a new nav bar in the <header> to look like this:



1. Notice the Login and Help links are at the right side of the screen. Adjust the browser window and notice how the <header> and <footer> remain at the top and bottom. Notice how the links stay at the left and right sides.
2. The HTML for the <header> nav bar looks like this:

<**header**>  
 <**h1**>This is the header</**h1**>  
 <**nav**>  
 <**div class="leftNav"**>  
 <**a href="#"**>First</**a**>  
 <**a href="#"**>Second</**a**>  
 <**a href="#"**>Third</**a**>  
 <**a href="#"**>Fourth</**a**>  
 </**div**>  
 <**div class="rightNav"**>  
 <**a href="#"**>Login</**a**>  
 <**a href="#"**>Help</**a**>  
 </**div**>  
  
 </**nav**>  
  
</**header**>

1. Notice the hierarchy. <header> has two children to allow the nav bar to be positioned below the header text. <header> <nav> has two children to allow some of the links to be on the left and the others on the right.
2. Add the CSS styles to create the image above using flex.

Congratulations. You have completed this lab.