**2018/9/7 Day14**

**1, when screen height/width is xx change xx**

Responsive Web Design Principles: Create a Media 
Query 
Media Queries are a new technique introduced in CSS3 that change the presentation of content 
based on different viewport sizes. The viewport is a user's visible area of a web page, and is 
different depending on the device used to access the site. 
Media Queries consist of a media type, and if that media type matches the type of device the 
document is displayed on, the styles are applied. You can have as many selectors and styles inside 
your media query as you want. 
Here's an example of a media query that returns the content when the device's width is less than 
or equal to IOOpx: 
@media (max-width: leopx) { CSS Rules s/ } 
and the following media query returns the content when the device's height is more than or 
equal to 350px: 
@media (min-height: 35ßpx) { /* CSS Rules x/ } 
Remember, the CSS inside the media query is applied only if the media type matches that of the 
device being used. 

Add a media query, so that the p tag has a font -size of IOpx when the device's height is less 
than or equal to 800px. 

<style> 
font-size: 2epx; 
Add media query below x/ 
@media (max-height: 80epx) { 
p { font-size: 18px; } 

**2, typography**

The four different viewport units are: 
vw: 10m.' would be 10% of the viewport's width. 
vh: 3vh would be ofthe viewport's height. 
vmln: 
vmax : 
7ßvmin would be 70% of the viewport's smaller dimension (height vs. width). 
lßevmax would be 100% of the viewport's bigger dimension (height vs. width). 

Set the width of the h2 tag to 80% of the viewport's width and the width of the paragraph as 75% 
of the viewport's smaller dimension. 

width : 
width : 
</style> 
gevw, 
75vmin; 
Ipsum< /h2> 
ipsum dalon sit amet, consectetur adipiscing elit. Vivamus quis tempus 
massa. Aenean eret nisl, gravida vel vestibulum cursus, interdum sit amet lectus . 
Sed sit amet quam nibh. Suspendisse quis tincidunt nulla. In hac habitasse platea 
dictumst. ut sit amet pretium nisl. Vivamus vel mi sem. Aenean sit amet 
consectetur sem. Suspendisse pretium, purus et gravida consequat, nunc ligula 
ultricies diam, at aliquet velit libero a 

**------CSS3 Flexbox**

1, display: flex;

this is used to adjust the position of content

**without display:flex**

body { 
font-family: 
header 
header 
Arial, 
sans 
-serif; 
4 
34 
. profile-thumbnail 
t,'idth: Sepx; 
height: Sepx; 
border-radius : 
header . profile 
Quincy Larson 
@ossia 
Follow 
I meet so many people who are in search of that one 
trick that will help them work smart. Even if you work 
smart, you still have to work hard. 
1 u PM - 12 Jan 2018 
4px , 
-name 
margin-left: 18px, 
header . follow-btn { 
margin: e a e auto; 
header . follow-btn button 
border: e; 
border-radius: 3px; 
padding: Spx; 
header h3, header 1-14 { 
margin: e; 
*inner p { 
margin-bottom: lapx; 
font-size: 2epx; 
*inner hr { 
margin: 2epx a; 
107 Retweets 
431 Likes 
Share Reftveet 
Like 

**add display:flex; to**

header, the header's .profile-name, the header's .follow-btn, the header's h3and h4, the footer, and the footer's .stats.

Quincy Larson@ossia 
38 
34 
header . follow-btn { 
display: flex; 
margin: e a e auto; 
header . follow-btn button 
border: e; 
border-radius: 3px; 
padding: Spx; 
header h3, header 1-14 
display: flex; 
margin: e; 
*inner p { 
margin-bottom: lapx; 
font-size: 2epx; 
*inner hr { 
margin: 2epx a; 
border-style: solid; 
opacity: 8.1; 
footer { 
display: flex; 
Follow 
I meet so many people who are in search of that one 
trick that will help them work smart. Even if you work 
smart, you still have to work hard. 
1 u PM - 12 Jan 2018 
107 Retweets 431 Likes 
Share 
Retweet 
Like 

2, flex-direction: row / row-reverse;

Creating a row will align the **children** horizontally, and creating a column will align the **children** vertically.

Forum 
7 
13 
18 
28 
23 
*box-container { 
display: flex; 
height: Seapx; 
*box-I { 
background -color : 
t,'idth: Sex, 
height: S"; 
*box-2 { 
background -color : 
t,'idth: Sex, 
height: S"; 
Curriculum 
dodgerblue; 
orangered; 
<div 
(div id="box-l"></div> 
(div 

7 
13 
18 
28 
23 
*box-container { 
display: flex; 
height: Seapx; 
flex-direction: 
*box-I { 
background -color : 
vddth: 58%, 
height: S"; 
*box-2 { 
background -color : 
vddth: 58%, 
height: S"; 
row-reverse 
dodgerblue; 
orangered; 
<div 
(div id="box-l"></div> 
(div 

2, flex-direction: column / column-reverse;

Forum 
<style> 
*box-container { 
display: flex; 
height: Seapx; 
*box-I { 
background -color : 
t,'idth: Sex, 
height: S"; 
*box-2 { 
background -color : 
t,'idth: Sex, 
height: S", 
Curriculum 
dodgerblue; 
orangered; 
<div 
(div id="box-l"></div> 
(div 

7 
13 
18 
28 
23 
*box-container { 
display: flex; 
height: Seapx; 
flex-direction : 
*box-I { 
background -color : 
t,'idth: Sex, 
height: S"; 
*box-2 { 
background -color: 
t,'idth: Sex, 
height: S", 
column; 
dodgerblue; 
orangered; 
<div 
(div id="box-l"></div> 
(div 

**BEFORE**

header . profile-name { 
display: flex; 
margin-left: 18px, 



**AFTER**

header . profile-name { 
display: flex; 
flex-direction: column; 
margin-left: 18px, 

Quincy Larson 
@ossia 

header . profile-name { 
display: flex; 
flex-direction: column-revers 
margin-left: 18px, 

@ossia 
Quincy Larson 

3, justify-content;

this will align items along with the main-axis, which is in a horizontal direction

Recall that setting a flex container as a row places the flex items side-by-side from left-to-right. A 
flex container set as a column places the flex items in a vertical stack from top-to-bottom. For 
each, the direction the flex items are arranged is called the main axis. For a row, this is a 
horizontal line that cuts through each item. And for a column, the main axis is a vertical line 
through the items. 
There are several options for how to space the flex items along the line that is the main axis. One 
of the most commonly used is justify-content: center; ,which aligns all the flex items to the 
center inside the flex container. Others options include: 
flex - start : aligns items to the start of the flex container. For a row, this pushes the items to 
the left of the container. For a column, this pushes the items to the top of the container. 
flex - end : aligns items to the end of the flex container. For a row, this pushes the items to 
the right of the container. For a column, this pushes the items to the bottom of the container. 
space -between : aligns items to the center ofthe main axis, with extra space placed between 
the items. The first and last items are pushed to the very edge of the flex container. For 
example, in a row the first item is against the left side of the container, the last item is against 
the right side of the container, then the other items between them are spaced evenly. 
space -around : similar to space- between but the first and last items are not locked to the 
edges of the container, the space is distributed around all the items 

4, align-items;

this will align items along with the cross-axis, which is in a vertical direction

Flex containers also have a cross axis which is the opposite of the main axis. For rows, the cross 
axis is vertical and for columns, the cross axis is horizontal. 
CSS offers the align -items property to align flex items along the cross axis. Fora row, it tells 
CSS how to push the items in the entire row up or down within the container. And for a column, 
how to push all the items left or right within the container. 
The different values available for align-items include: 
flex - start : aligns items to the start of the flex container. For rows, this aligns items to the 
top of the container. For columns, this aligns items to the left of the container. 
flex - end : aligns items to the end of the flex container. For rows, this aligns items to the 
bottom of the container. For columns, this aligns items to the right of the container. 
• center : align items to the center. For rows, this vertically aligns items (equal space above 
and below the items). For columns, this horizontally aligns them (equal space to the left and 
right of the items). 
• stretch : stretch the items to fill the flex container. For example, rows items are stretched to 
fill the flex container top-to-bottom. 
baseline : align items to their baselines. Baseline is a text concept, think of it as the line that 
the letters sit on. 

5, flex-wrap:

to wrap items

However, using the flex -wrap property, it tells CSS to wrap items. This means extra items move 
into a new row or column. The break point of where the wrapping happens depends on the size of 
the items and the size of the container. 
CSS also has options for the direction of the wrap: 
nourap : this is the default setting, and does not wrap items. 
• wrap : wraps items from left-to-right ifthey are in a row, or top-to-bottom if they are in a 
column. 
wrap - reverse : wraps items from bottom-to-top if they are in a row, or left-to-right if they are 
in a column. 

*box-container { 
background: gray; 
display: flex; 
height: 188%; 
*box-I { 
background -color : 
t,'idth: 25%, 
height: S", 
*box-2 { 
background -color: 
t,'idth: 25%, 
height: S", 
*box-3 { 
background -color: 
t,'idth: 25%, 
height: S", 
*box-4 { 
background -color : 
t,'idth: 25%, 
height: S", 
*box-5 { 
background -color : 
t,'idth: 25%, 
height: S"; 
dodgerblue; 
orangered; 
violet; 
yellow; 
green; 

8 
14 
*box-container { 
background: gray; 
display: flex; 
height: 188%; 
*box-I { 
background -color : 
t,'idth: 25%, 
height: S", 
*box-2 { 
background -color : 
t,'idth: 25%, 
height: S"; 
*box-3 { 
background -color : 
t,'idth: 25%, 
height: S"; 
*box-4 { 
background -color : 
t,'idth: 25%, 
height: S", 
*box-5 { 
background -color : 
t,'idth: 25%, 
height: S", 
*box-6 { 
background -color : 
dodgerblue; 
orangered; 
violet; 
yellow; 
green; 
black; 
running test 

6, item size: flex-shrink (to shrink), flex-grow(to expand), flex-basis(to initialize)

快捷办法 flex: grow, shrink, basis

For example, if one item has a flex-shrinkvalue of 1 and the other has a flex-shrinkvalue of 3, the one with the value of 3 will shrink three times as much as the other.

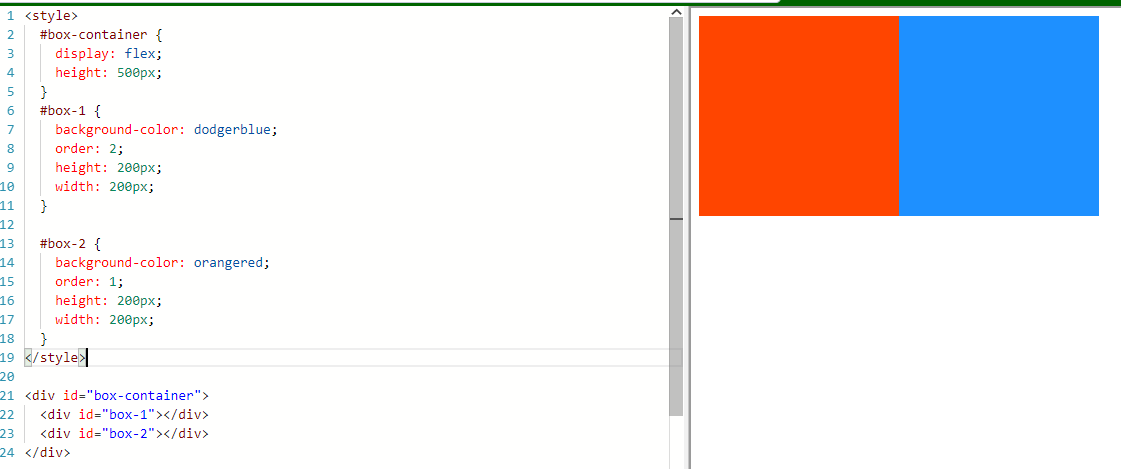
6 
13 
19 
*box-container { 
display: flex; 
height: Seapx; 
*box-I { 
background -color : 
t,'idth: lee%; 
height: 2eapx; 
*box-2 { 
background - color : 
t,'idth: lee%; 
height: 2eapx; 
dodgerblue; 
orangered; 
21 <div id=" box-container" > 
(div id="box-l"></div> 
(div 
24 

Forum 
6 
*box-container { 
display: flex; 
height: Seapx; 
*box-I { 
background - color: 
t,'idth: lee%; 
height: 2eapx; 
flex-shrink: 1, 
*box-2 { 
background - color: 
t,'idth: lee%; 
height: 2eapx; 
Flex-shrink: 
Curriculum 
dodgerblue; 
orangered; 
<div 
(div id="box-l"></div> 
(div 

7, order: value

default的顺序是按HTML代码来的，但是用order，order: 1 就排在 order:2 前

橘色方块HTML代码滞后，本来应该在右边。但是因为是order:1 就排在了order:2 的前面了



8， aligh-self: 单独调节一个itme位置

之前的都是会一起调节

