Разработка пользовательского веб-интерфейса

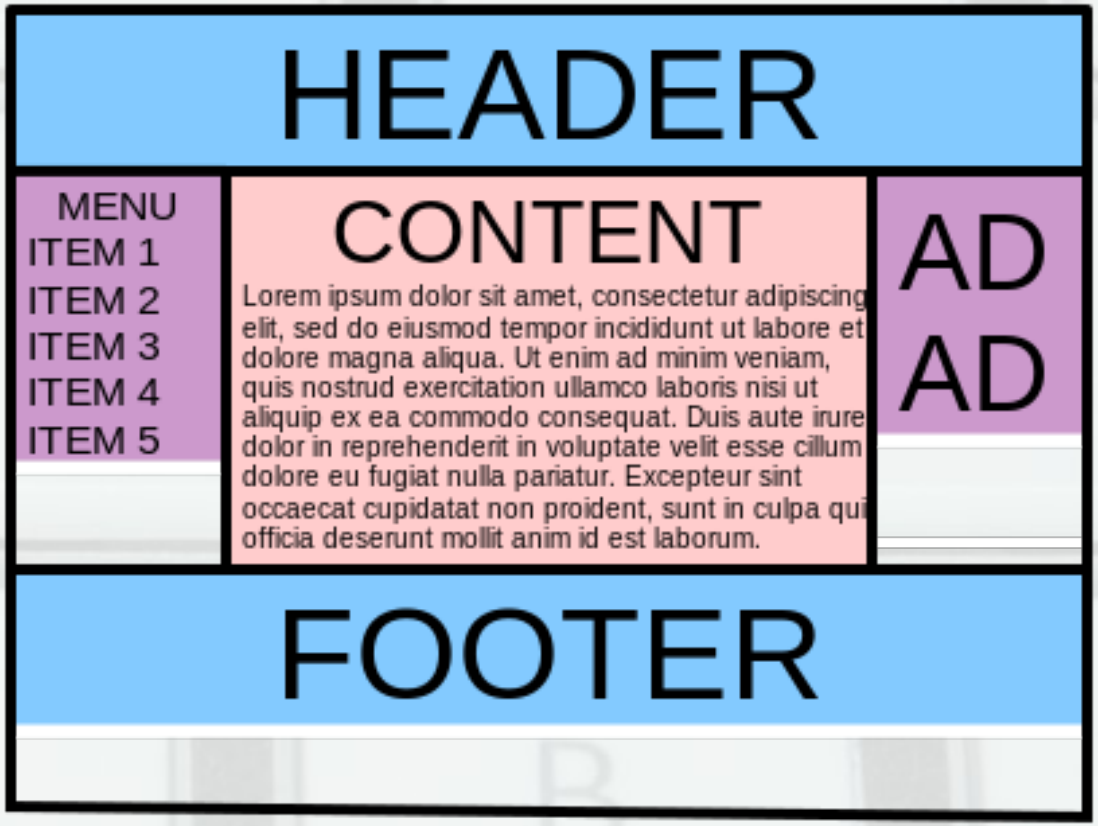
Адаптивный веб-дизайн, CSS3

Сергей Геннадьевич Синица

КубГУ, 2021

sin@kubsu.ru

Holy Grail!


HEADER
MENU
ITEM 1
ITEM 2
ITEM 3
ITEM 4
ITEM 5
CONTENT
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
AD
AD
FOOTER

CSS3 Flex Box

– Располагать элементы слева направо, справа налево, сверху вниз или снизу вверх.

– Переопределять порядок отображения элементов в потоке.

– Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство.

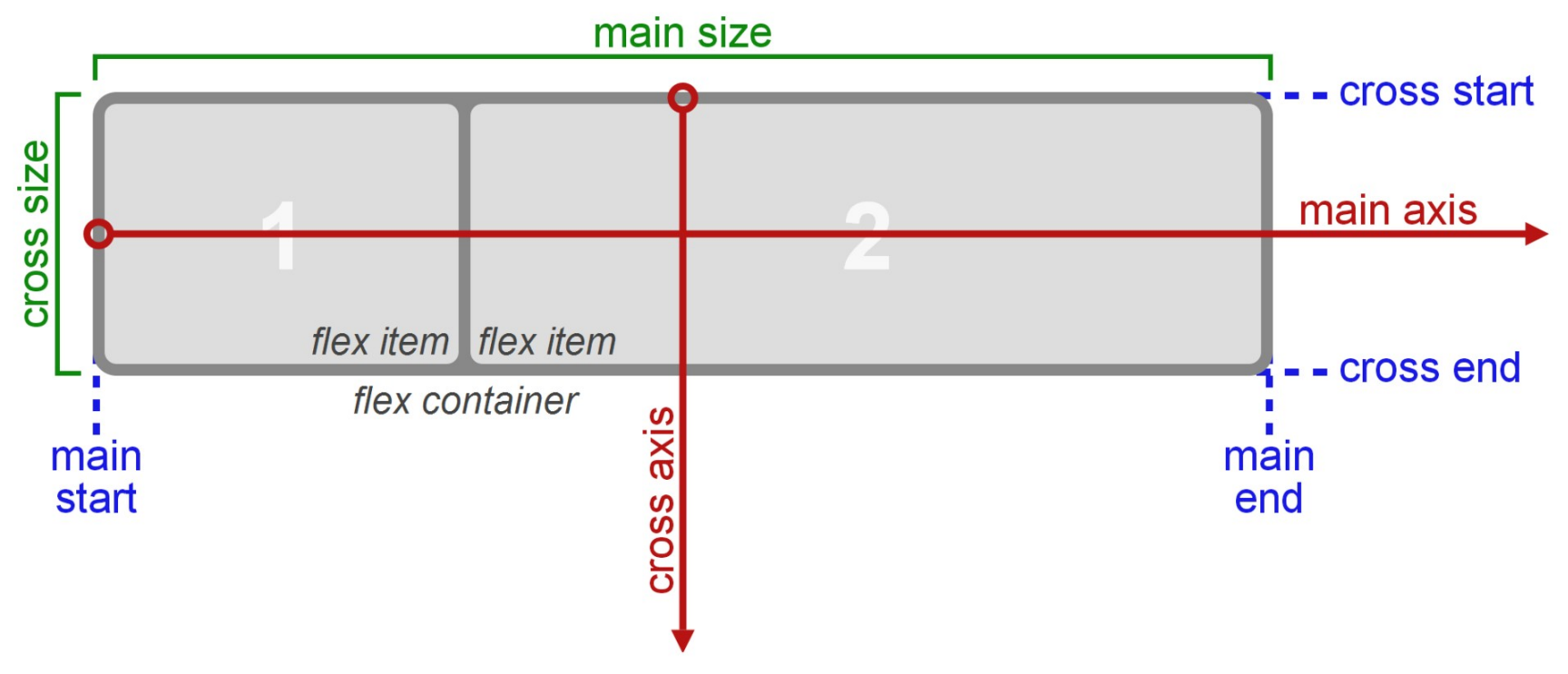
– Выравнивать элементы горизонтально и вертикально относительно оси.

– Создавать колонки одинаковой высоты, аналогично ячейкам таблицы.

– Создавать прижатый к низу окна браузера подвал сайта.

– https://caniuse.com/#feat=flexbox 99%

CSS3 Flex Box



https://html5book.ru/css3-flexbox/

Для flex-элементов блокируется их значение свойства display. Значение display: inline-block; и display: table-cell; вычисляется в display: block;.

Пустое пространство между элементами исчезает: оно не становится своим собственным flex-элементом, даже если межэлементный текст обернут в анонимный flex-элемент. Для содержимого анонимного flex-элемента невозможно задать собственные стили, но оно будет наследовать их (например, параметры шрифта) от flex-контейнера.

Абсолютно позиционированный flex-элемент не участвует в компоновке гибкого макета.

Поля margin соседних flex-элементов не схлопываются.

Процентные значения margin и padding вычисляются от внутреннего размера содержащего их блока.

margin: auto; расширяются, поглощая дополнительное пространство в соответствующем измерении. Их можно использовать для выравнивания или раздвигания смежных flex-элементов.

Автоматический минимальный размер flex-элементов по умолчанию является минимальным размером его содержимого, то есть min-width: auto;. Для контейнеров с прокруткой автоматический минимальный размер обычно равен нулю.

flex-direction

row Значение по умолчанию, слева направо (в rtl справа налево). Flex-элементы выкладываются в строку. Начало (main-start) и конец (main-end) направления главной оси соответствуют началу (inline-start) и концу (inline-end) инлайн-оси (inline-axis).

row-reverse Направление справа налево (в rtl слева направо). Flex-элементы выкладываются в строку относительно правого края контейнера (в rtl — левого).

column Направление сверху вниз. Flex-элементы выкладываются в колонку.

column-reverse Колонка с элементами в обратном порядке, снизу вверх.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

flex-wrap

nowrap Значение по умолчанию. Flex-элементы не переносятся, а располагаются в одну линию слева направо (в rtl справа налево).

wrap Flex-элементы переносятся, располагаясь в несколько горизонтальных рядов (если не помещаются в один ряд) в направлении слева направо (в rtl справа налево).

wrap-reverse Flex-элементы переносятся на новые линии, располагаясь в обратном порядке слева-направо, при этом перенос происходит снизу вверх.

initial Устанавливает значение свойства в значение по умолчанию.

inherit Наследует значение свойства от родительского элемента.

Краткая запись:
flex-flow: row nowrap;

order

Свойство определяет порядок, в котором flex-элементы отображаются и располагаются внутри flex-контейнера. Применяется к flex-элементам. Свойство не наследуется.

Первоначально все flex-элементы имеют order: 0;. При указании значения от -1 для элемента он перемещается в начало сроки, значение 1 — в конец. Если несколько flex-элементов имеют одинаковое значение order, они будут отображаться в соответствии с исходным порядком.

flex

Свойство является сокращённой записью свойств flex-grow, flex-shrink и flex-basis. Значение по умолчанию: flex: 0 1 auto;. Можно указывать как одно, так и все три значения свойств.

Flex-контейнер распределяет свободное пространство между своими дочерними элементами (пропорционально их коэффициенту flex-grow) для заполнения контейнера или сжимает их (пропорционально их коэффициенту flex-shrink), чтобы предотвратить переполнение.

Flex-элемент будет полностью «негибок», если его значения flex-grow и flex-shrink равны нулю, и «гибкий» в противном случае.

flex-grow определяет коэффициент роста одного flex-элемента относительно других flex-элементов в flex-контейнере при распределении положительного свободного пространства. Если сумма значений flex-grow flex-элементов в строке меньше 1, они занимают менее 100% свободного пространства. Свойство не наследуется.

justify-content

flex-start Значение по умолчанию. Flex-элементы выкладываются в направлении, идущем от начальной линии flex-контейнера.

flex-end Flex-элементы выкладываются в направлении, идущем от конечной линии flex-контейнера.

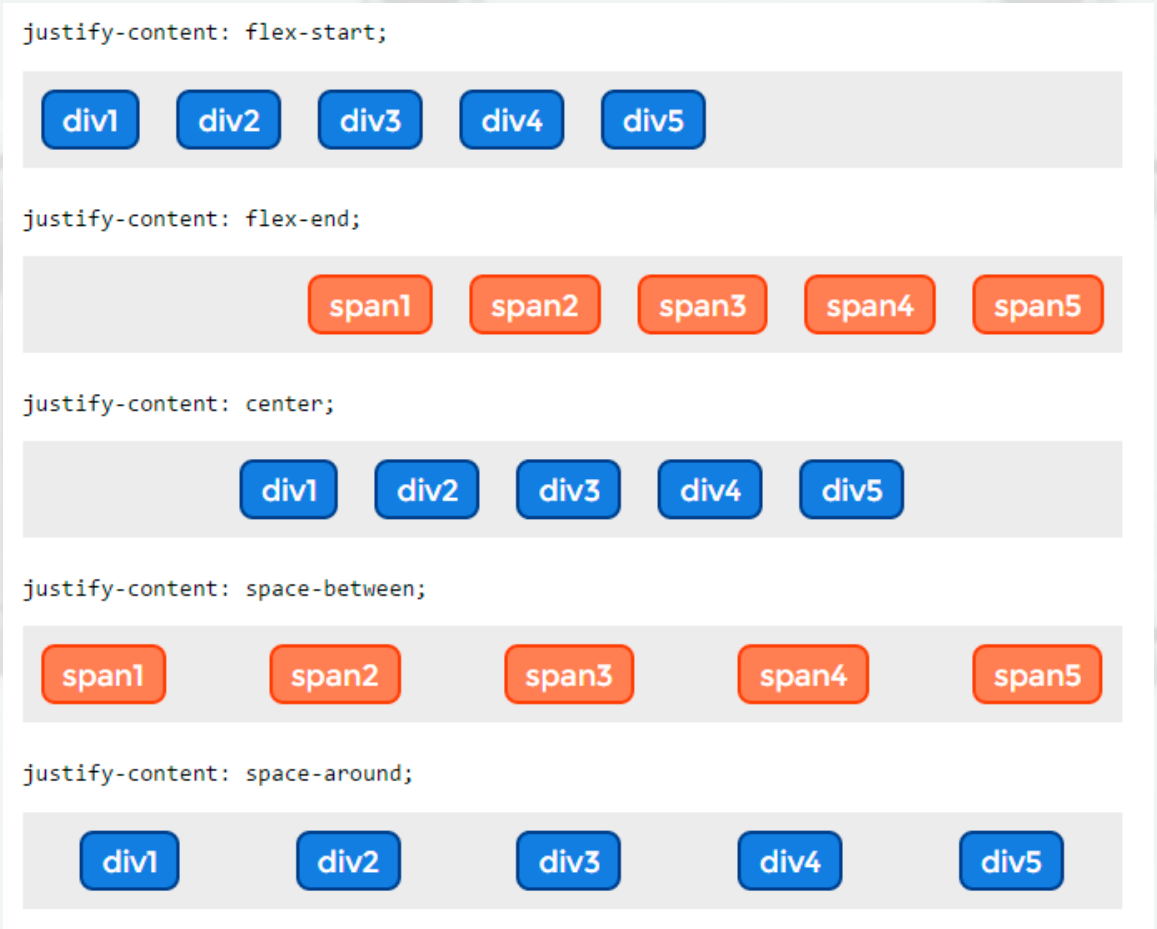
center Flex-элементы выравниваются по центру flex-контейнера.

space-between Flex-элементы равномерно распределяются по линии. Первый flex-элемент помещается вровень с краем начальной линии, последний

flex-элемент — вровень с краем конечной линии, а остальные flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым. Если оставшееся свободное пространство отрицательно или в строке присутствует только один flex-элемент, это значение идентично параметру flex-start.

space-around Flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между flex-элементами.

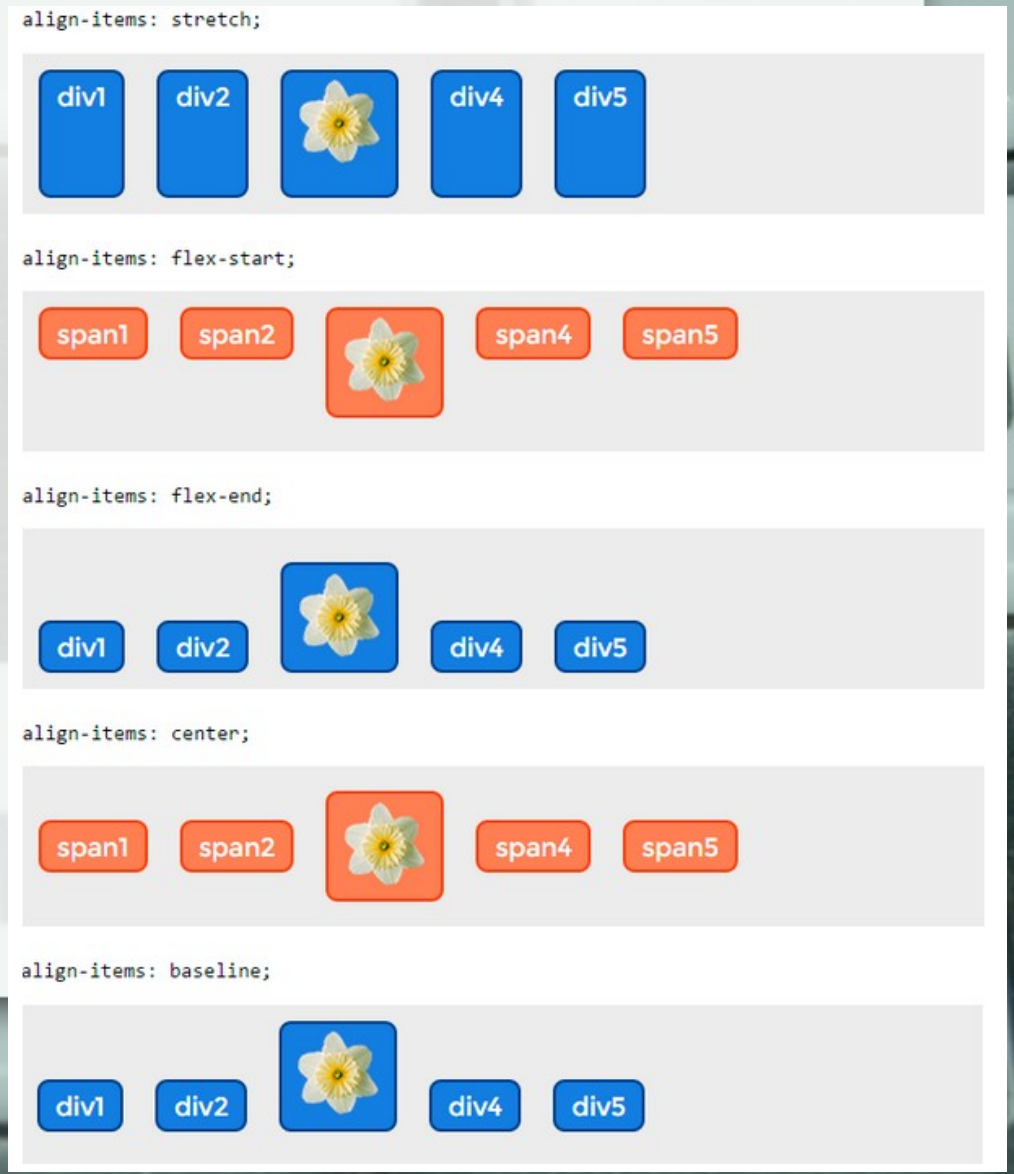
justify-content



Источник https://html5book.ru/css3-flexbox/

align-items и align-self

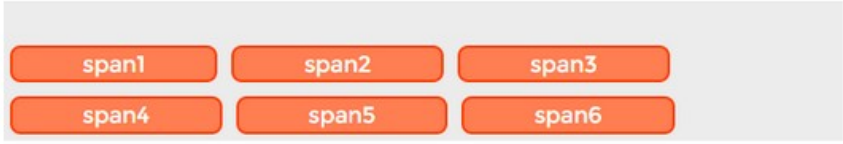
Flex-элементы можно выравнивать по поперечной оси текущей строки flex-контейнера. align-items устанавливает выравнивание для всех элементов flex-контейнера, включая анонимные flex-элементы. align-self позволяет переопределить это выравнивание для отдельных flex-элементов. Если любое из поперечных margin flex-элемента имеет значение auto, align-self не имеет никакого влияния. n-self



align-content

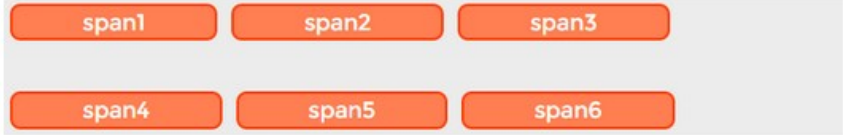
```
align-content: flex-start;
```


```
align-content: flex-end;
```



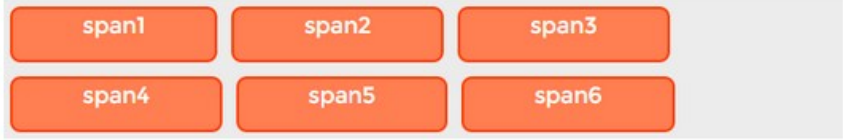

```
align-content: center;
```


```
align-content: space-between;
```




```
align-content: space-around;
```

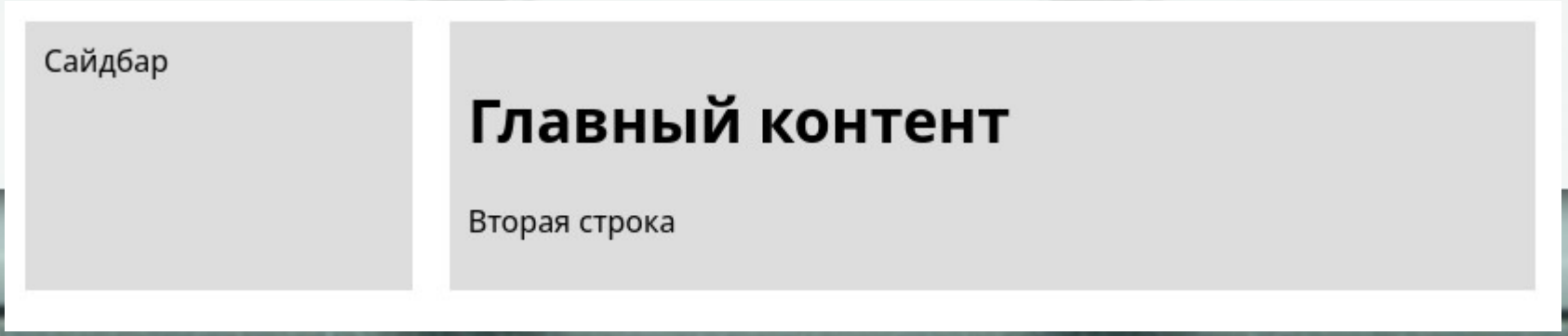

```
align-content: stretch;
```



```
    <style>
#main-aside-wrapper {
  flex-direction: row;
  display: flex;
}
#main-aside-wrapper main, #main-aside-wrapper aside {
  background-color: #ddd;
  margin: 10px;
  padding: 10px;
}
#main-aside-wrapper aside {
  flex: 1;
  order: 1;
}
#main-aside-wrapper main {
  flex: 3;
  order: 2;
}
   </style>
```

```
<body>
   <div id="main-aside-wrapper">
     <main>
       <h1>Главный контент</h1>
       <p>Вторая строка</p>
     </main>
     <aside>Сайдбар</aside>
   </div>
 </body>

</html>
```



CSS3 Grid Layout

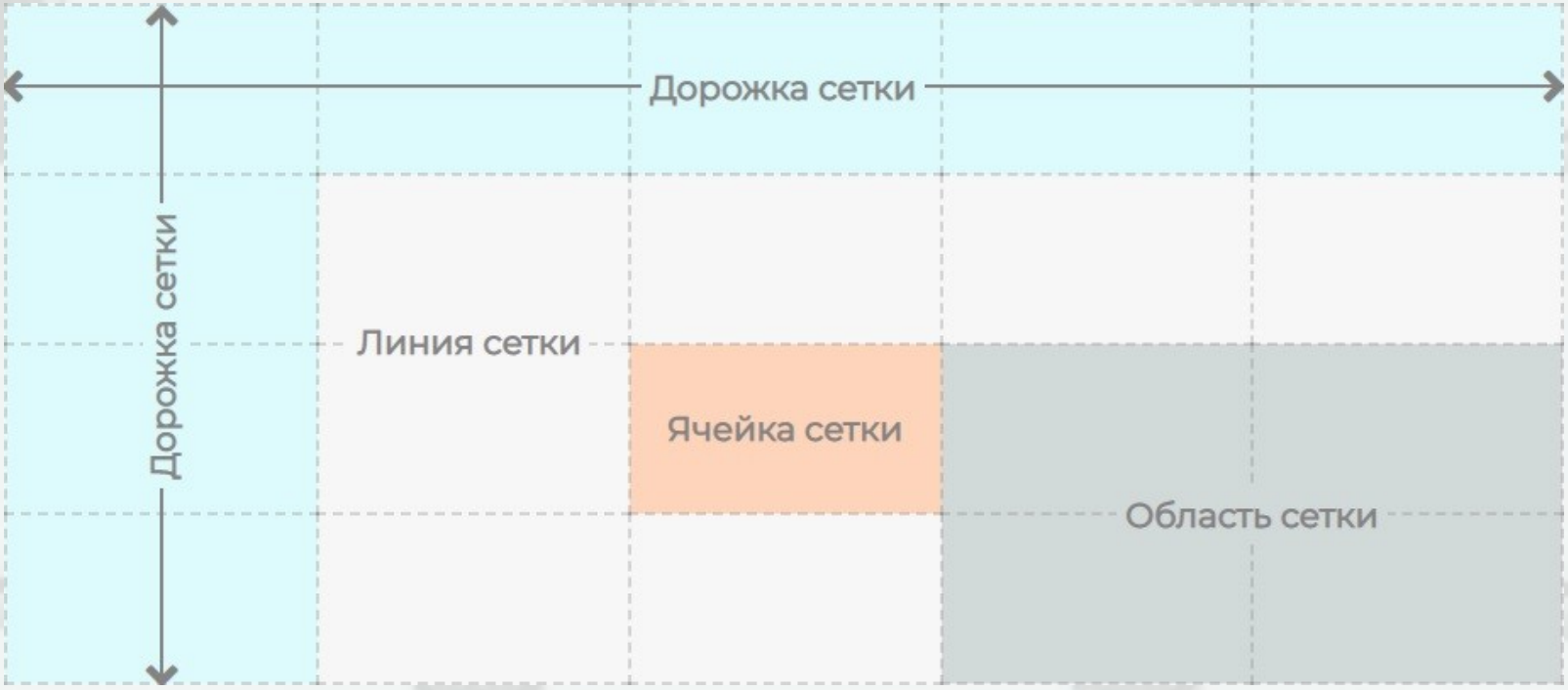
– Располагать элементы в двумерной сетке.

– Произвольно выставлять положение элемента по сетке.

– Комбинировать фиксированные и относительные размеры.

– https://caniuse.com/#feat=css-grid 93%

CSS3 Grid Layout



https://html5book.ru/css-grid/

```
.grid {
  display: grid;
  grid: 1fr 25% 30px / 40% 1fr; /* ряды / колонки */
  grid-gap: 1em;
  height: 200px;
}

<div class="grid">
  <div class="item">item 1</div>
  <div class="item">item 2</div>
  <div class="item">item 3</div>
  <div class="item">item 4</div>
  <div class="item">item 5</div>
  <div class="item">item 6</div>
</div>
```

*1fr – fraction of leftover space

Adaptive vs Responsive

Адаптивный дизайн – подстраивается под пользователя.

Отзывчивый дизайн (Ethan Marcotte 2009) – адаптивная верстка CSS 3 с использованием:

- Fluid Grids (“резиновая верстка”, сетка в %)
- Flexible Images (max-width: 100%;)
- Media Queries

https://alistapart.com/article/responsive-web-design/

https://alistapart.com/article/fluidgrids/

http://unstoppablerobotninja.com/entry/fluid-images

https://960.gs

Поиск

If you want a responsive grid framework, check out Unsemantic.com

Twitter @NATHANSMITH

960 GRID SYSTEM

Download - CSS, sketch paper, and templates for: Acorn, Fireworks, Flash, InDesign, GIMP, Inkscape, Illustrator, OmniGraffle, Photoshop, QuarkXPress, Visio, Exp Design. Repository at GitHub.

Big ol' DOWNLOAD button :)

the changelog INTERVIEW ABOUT 960.gs

VIEW SLIDES ABOUT THE 960 GRID SYSTEM

ADAPT.JS – ADAPTIVE CSS

CUSTOM CSS GENERATOR

GRID OVERLAY BOOKMARK

Essence

The 960 Grid System is an effort to streamline web development workflow by providing commonly used dimensions, based on a width of 960 pixels. There are two variants: 12 and 16 columns, which can be used separately or in tandem. Read more.

Dimensions

The 12-column grid is divided into portions that are 60 pixels wide. The 16-column grid consists of 40 pixel increments. Each column has 10 pixels of margin on the left and right, which create 20 pixel wide gutters between columns. View demo.

Purpose

The premise of the system is ideally suited to rapid prototyping, but it would work equally well when integrated into a production environment. There are printable sketch sheets, design layouts, and a CSS file that have identical measurements.


```
h1 {
    margin-left: 14.575%;     /* 144px / 988px = 0.14575 */
    width: 70.85%;            /* 700px / 988px = 0.7085 */
}
```

CSS 2.1 Media Types

```
<link rel="stylesheet" type="text/css" href="core.css"
  media="screen" />

<link rel="stylesheet" type="text/css" href="print.css"
  media="print" />
```

CSS 3 Media Queries

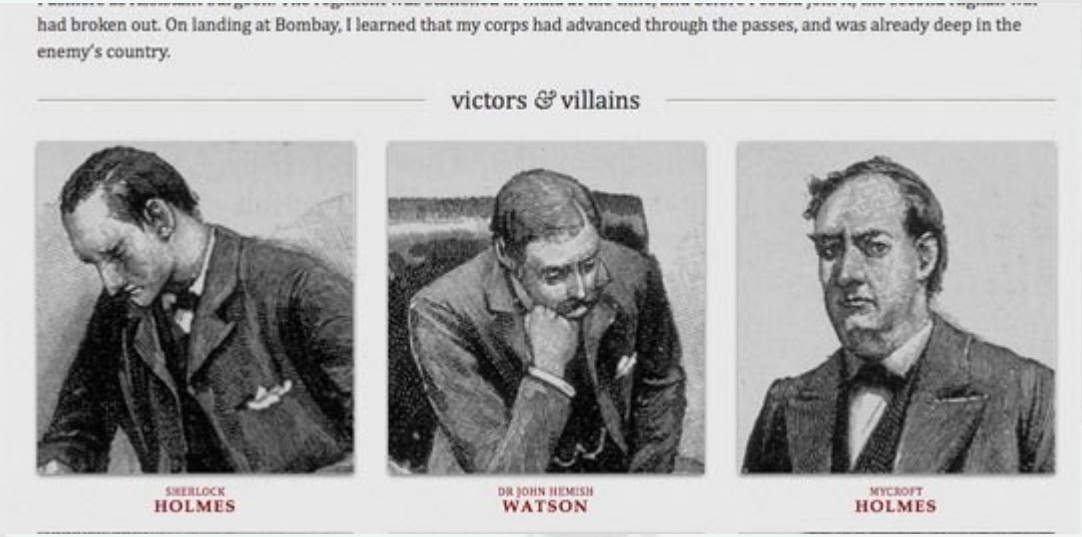
```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px)"
  href="shetland.css" />
```

CSS 3 Media Queries

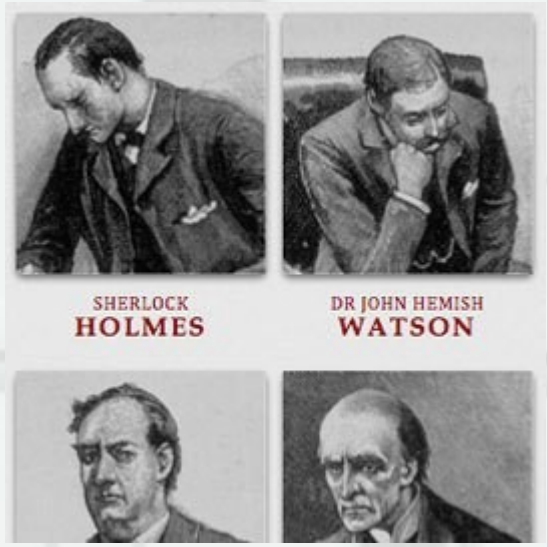
```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px) and (resolution: 163dpi)"
  href="shetland.css" />
```

CSS 3 Media Queries

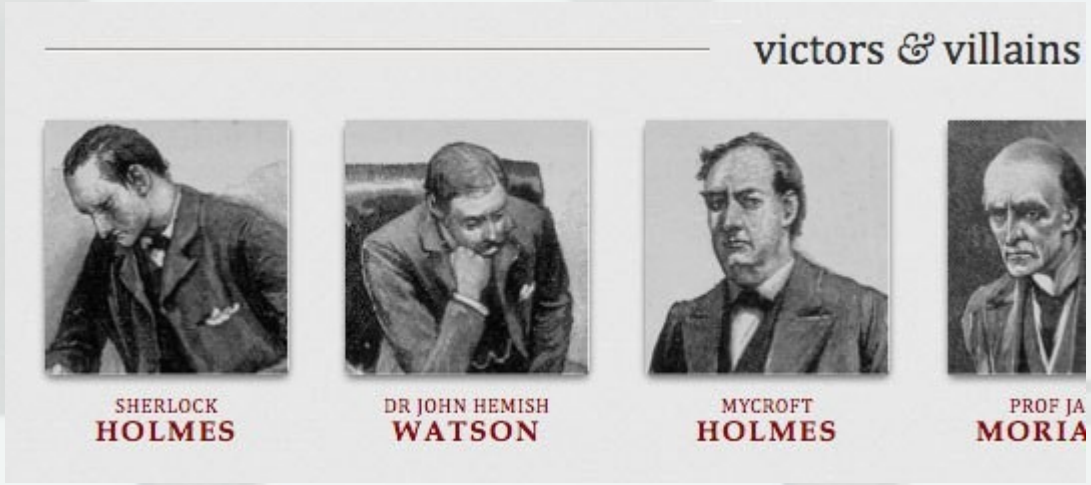
```
@media screen and (max-device-width: 480px) {
  .column {
    float: none;
  }
}
```




```
.figure {
  float: left;
  margin: 0 3.3175355450236966% 1.5em 0;   /* 21px / 633px */
  width: 31.121642969984202211%;           /* 197px / 633px */
}
li#f-mycroft, li#f-winter {
  margin-right: 0;
}
```



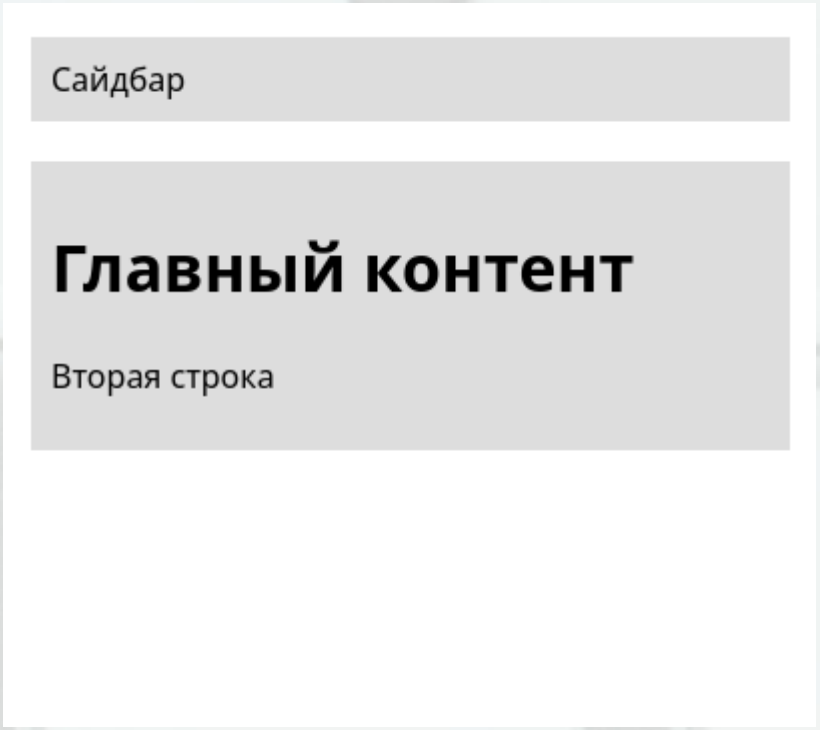

```
@media screen and (max-width: 400px) {
  .figure,  li#f-mycroft {
    margin-right: 3.31753554502369668%;    /* 21px / 633px */
    width: 48.341232227488151658%;         /* 306px / 633px */
  }
 li#f-watson,  li#f-moriarty {
    margin-right: 0;
  }
}
```

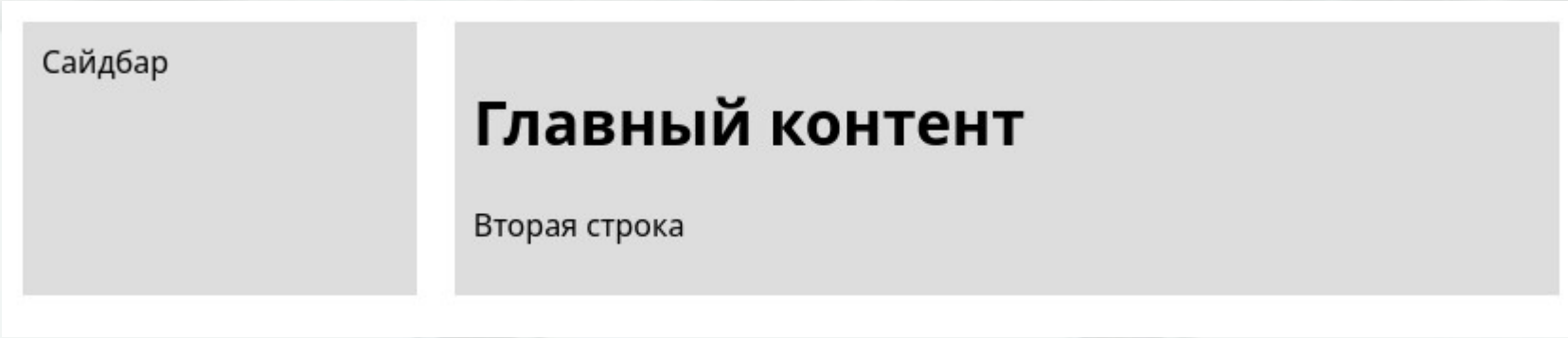



```
@media screen and (min-width: 1300px) {
  .figure,
  li#f-mycroft {
    margin-right: 3.31753554502369682%;    /* 21px / 633px */
    width: 13.902053712480252764%;         /* 88px / 633px */
  }
}
```

Контент первый в потоке!

```
<div id="main-aside-wrapper">
  <main>
    <h1>Главный контент</h1>
    <p>Вторая строка</p>
  </main>
  <aside>Сайдбар</aside>
</div>
```





Mobile First!

```
#main-aside-wrapper {
  display: flex;
  flex-direction: column;
}
#main-aside-wrapper main, #main-
aside-wrapper aside {
  background-color: #ddd;
  margin: 10px;
  padding: 10px;
}
#main-aside-wrapper aside {
  flex: 1;
  order: 1;
}
#main-aside-wrapper main {
  flex: 3;
  order: 2;
}

@media screen and (min-width:
800px) {
  #main-aside-wrapper {
    flex-direction: row;
  }
}
```

```
<div id="main-aside-wrapper">
  <main>
    <h1>Главный контент</h1>
    <p>Вторая строка</p>
  </main>
  <aside>Сайдбар</aside>
</div>
```

Viewport

Viewport это часть окна браузера, где отображается страница.

Если страница больше, то появляется скролл.

На смартфонах физическое разрешение вьюпорта может отличаться от логического, например некоторые телефоны рендерят страницу 980px ширины и ресайзят в экран 640px чтобы отобразить не оптимизированные под мобайл сайты без скролла.

Чтобы работали брейкпоинты на резрешения менее 980px такой ресайз отключаем, добавляя в секцию head приравнивание ширины вьюпорта CSS-ширине экрана:

<meta name="viewport" content="width=device-width, initial-scale=1">

Initial-scale это начальный уровень зума.

Подробнее:
https://developer.mozilla.org/en-US/docs/Web/HTML/Viewport_meta_tag

CSS-пиксели – не пиксели?

CSS-пиксели, в которых задается размер всех элементов, меньше реального разрешения устройств!

Гаджет	Вьюпорт	Разрешение экрана
IPhone 12	390 x 844	1170 x 2532
iPad Air 1 & 2	768 x 1024	1536 x 2048
Samsung Galaxy S9+	360 x 740	1440 x 2960

Вьюпорты и физическое разрешение на разных смартфонах:
https://viewportsizer.com/devices/

Прощай %, привет vh и vw!

% отсчитывается от родительского элемента

vh и vw отсчитываются от размеров окна

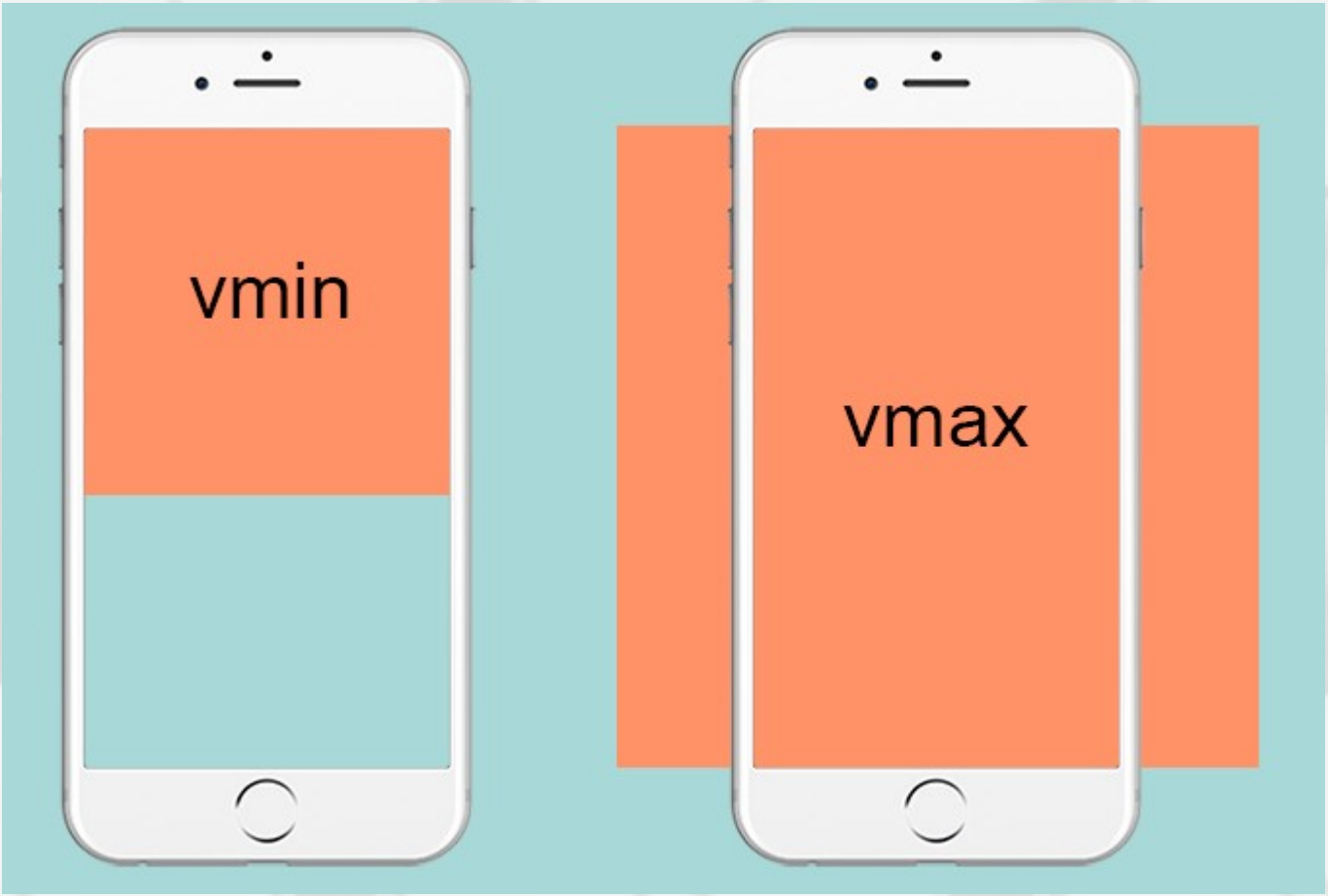
100vw на десктопе больше области просмотра из-за скролбара, поэтому для картинок используем 100%

При ширине окна 1000px

1vw = 10px

```
.fullscreen-bg {
  background: url(image.jpg);
  background-position: center;
  background-size: cover;
  width: 100%;
  height: 100vh;
}
```

vmin и vmax



Минимум или максимум высоты или ширины