Вёрстка

Татаринова А.Г.
к.т.н., доцент кафедры ПМИ

2025

Вёрстка

Вёрстка веб-страниц — создание структуры отображения в браузере веб-страницы на основе HTML разметки и CSS стилей

(!) нужно учитывать разницу отображения элементов в различных браузерах и разницу в размерах рабочего пространства устройств

(!) следует принимать во внимание ограничения HTML и CSS

Виды вёрстки

- Табличная
- Блочная
- Семантическая
- Адаптивная

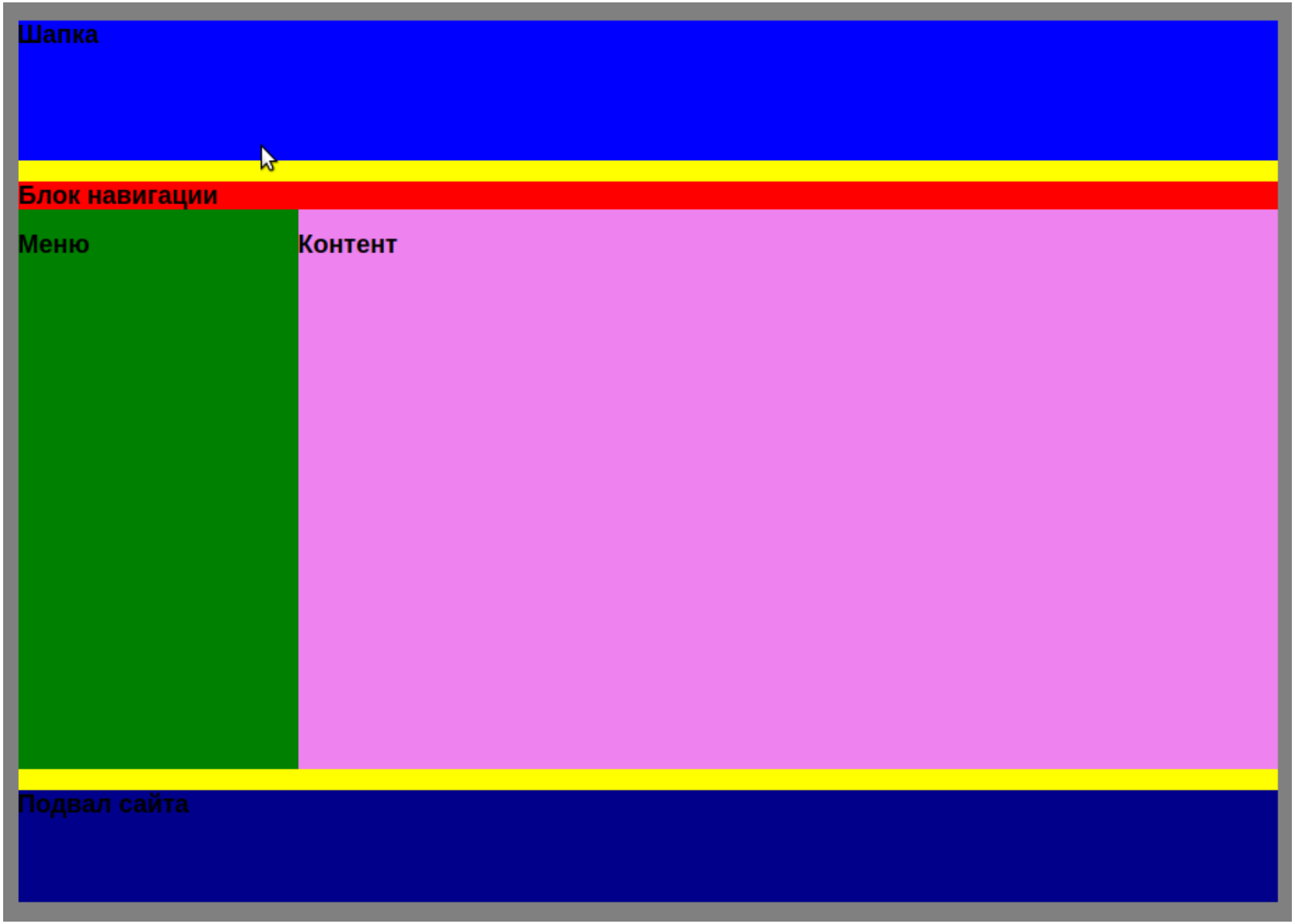
Табличная вёрстка

- Использование таблиц с невидимой границей
- Такая таблица представляет модульную сетку, в которой удобно размещать отдельные элементы веб-страницы
- (+): простота и быстрота верстки, а также корректное отображение в различных браузерах
- (-): содержимое страницы, сверстанной на основе таблиц, не будет отображено до тех пор, пока не загрузятся все данные страницы
- (-): получается объемный код
- (-): трудоемко перестраивать и обновлять веб-страницу

Блочная вёрстка

- Основным элементом является тег ** **
  Иногда участок кода, отделенный этим тегом, называется слоем
- Отделение стиля элементов от кода HTML
- Лучшая индексация поисковиками
- Более высокая скорость загрузки страницы, состоящей от взаимно независимых элементов
- Возможность создания визуальных эффектов (выпадающих меню, списков, всплывающих подсказок)
- Основным недостатком блочной верстки является некая «двусмысленность» понимания ее кода различными браузерами

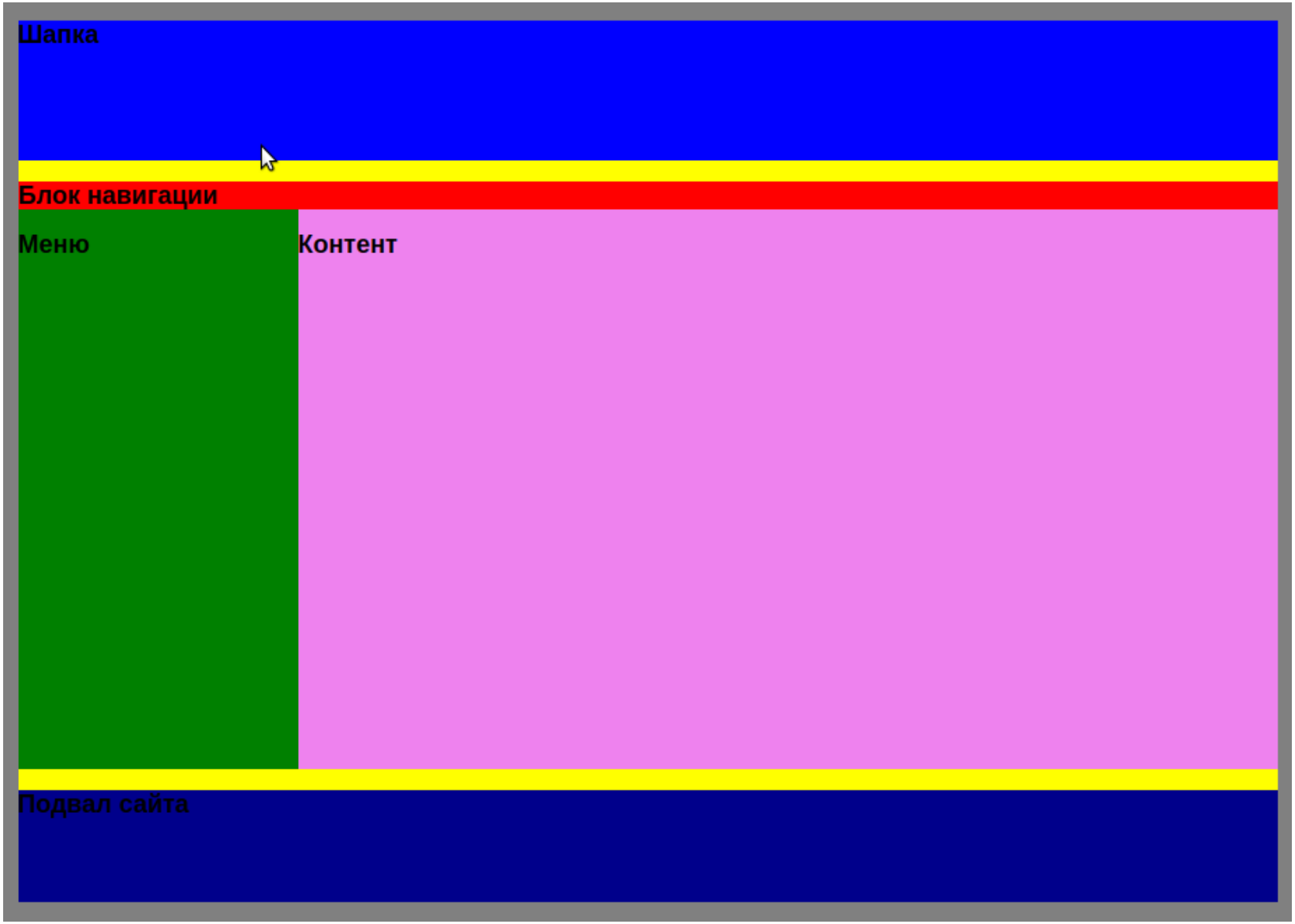
Пример 1


Шапка
Блок навигации
Меню
Контент
Подвал сайта

Семантическая вёрстка

- Предполагает, что страницу можно разбить на логические блоки, выделив заголовок **<header>**, подвал **<footer>**, основной контент **<main>** и другие элементы
- Позволяет лучше структурировать документ, а также облегчает работу поисковым роботам, которые индексируют страницы, и при правильной разметке красиво отображают их в поисковой выдаче

Пример 2


Шапка
Блок навигации
Меню
Контент
Подвал сайта

Блочные и строчные элементы

- Блочные элементы обычно используются для создания структуру веб-страницы
- Строчные элементы обычно используются для форматирования текстовых фрагментов (кроме `<img>` и `<area>`)
- Актуально до HTML 4.1
- В HTML5 заменены более сложным набором категорий контента, согласно которым каждый HTML-элемент должен следовать правилам, определяющим, какой контент для него допустим

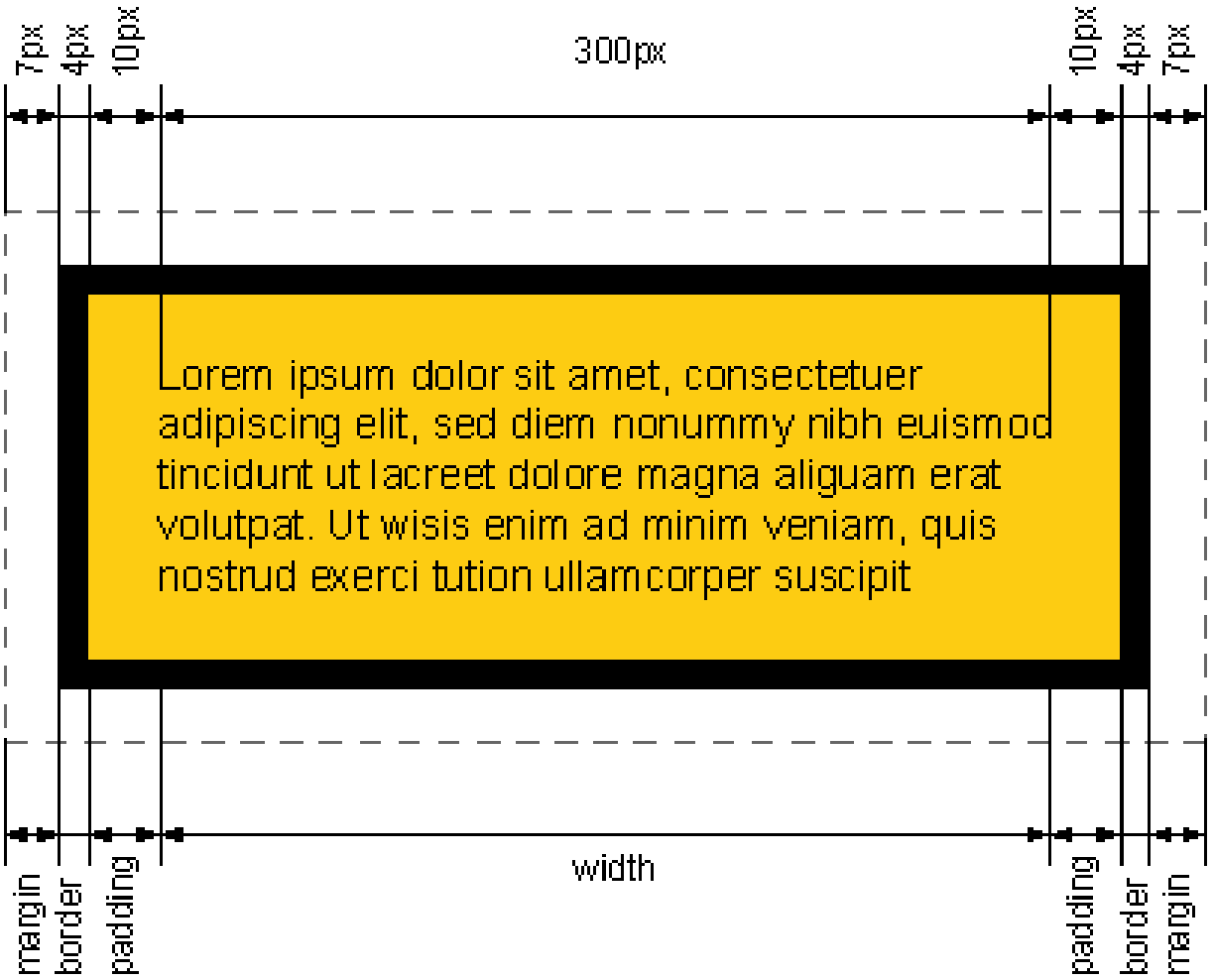
Блочные элементы

- Блочным называется элемент, который определяет области или части HTML документа
- Такой элемент занимает всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки
- К блочным элементам относится ** **, а также **<h1>**, **<p>**
- (!) Допускается вкладывать один блочный элемент внутрь другого, а также размещать внутри них строчные элементы (например, ****)
- (!) Запрещено добавлять внутрь строчных элементов блочные

Ширина блочных элементов

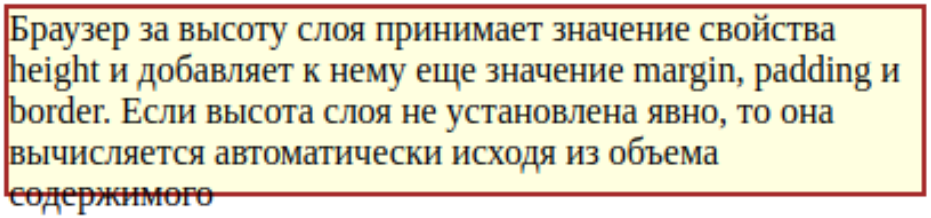
- По умолчанию ширина блока вычисляется автоматически и занимает все доступное пространство
- Например, если тег `<div>` в коде документа присутствует один, то он занимает всю свободную ширину окна браузера и ширина блока будет равна 100%
- Если поместить один тег `<div>` внутрь другого, то ширина внутреннего тега начинает исчисляться относительно его родителя, т.е. внешнего контейнера

Ширина блочных элементов



Высота блочных элементов

- Браузер за высоту элемента принимает значение свойства **`height`** и добавляет к нему еще значение **`margin`**, **`padding`** и **`border`**
- Если высота элемента не установлена явно, то она вычисляется автоматически исходя из объема содержимого

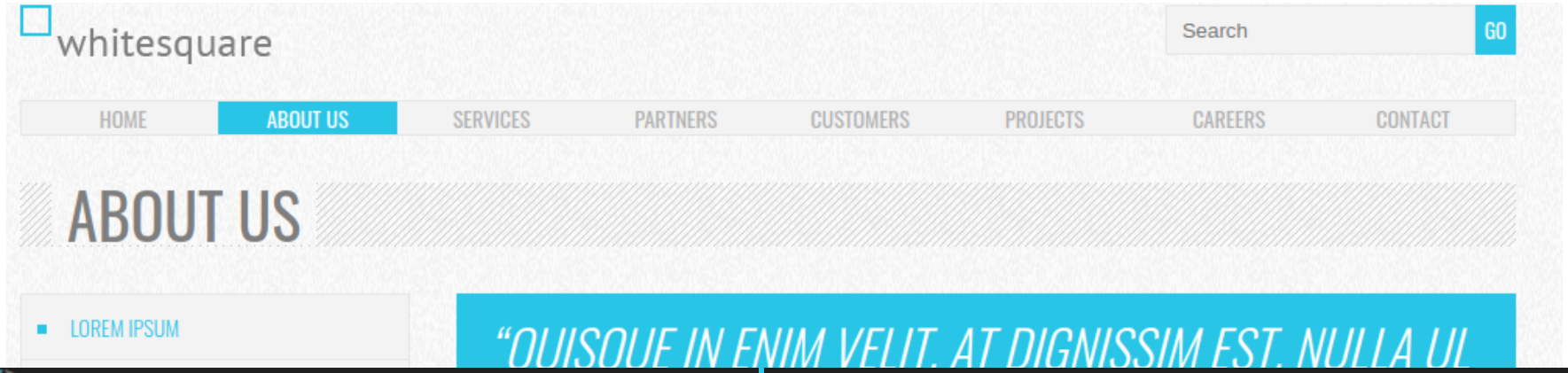


Пример 3

Строчные элементы

- Строчными называются такие элементы веб-страницы, которые являются непосредственной частью другого элемента, например, текстового абзаца
- К строчным элементам относятся теги ****, **<a>** и др.
- В основном они используются для изменения вида текста или его логического выделения

Пример центрирования




```html
<nav>
    <ul class="top-menu">
        <li><a href="/home/">HOME</a></li>
        <li class="active">ABOUT US</li>
        <li><a href="/services/">SERVICES</a></li>
        <li><a href="/partners/">PARTNERS</a></li>
        <li><a href="/customers/">CUSTOMERS</a></li>
        <li><a href="/projects/">PROJECTS</a></li>
        <li><a href="/careers/">CAREERS</a></li>
        <li><a href="/contact/">CONTACT</a></li>
    </ul>
</nav>
```

```css
.top-menu{
    text-align: center; /*Центрирование*/
}

.top-menu li {
    display: inline-block; /*Для центрирования*/
    padding: 0px 0px;
    margin: 0;
    text-align: center; /*Центрирование*/
    width: 12%; /*Ширина*/
}
```

Пример 4

Плавающие элементы

- Элементы, которые обтекаются по контуру другими объектами веб-страницы, например, текстом
- Используется стилевое свойство **`float`**
- **`float: left | right | none | inherit;`**

The Normal Document Flow

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

Right-Floating Image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

```
<img src="http://lorempixum.com/200/200/" />
<p>Lorem ipsum...</p>
```

```
img {
  float: right;
  margin: 20px;
}
```

Плавающие элементы

При применении свойства `float` **происходит следующее**:

- Элемент позиционируется как обычно, а затем вынимается из “документа” и сдвигается влево (для left) или вправо (для right) до того как коснётся либо границы родителя, либо другого элемента с float
- Если пространства по горизонтали не хватает для того, чтобы вместить элемент, то он сдвигается вниз до тех пор, пока не начнёт помещаться
- Другие непозиционированные блочные элементы без float ведут себя так, как будто элемента с float нет
- Строки (inline-элементы), напротив, «знают» о float и обтекают элемент по сторонам

Особенности:

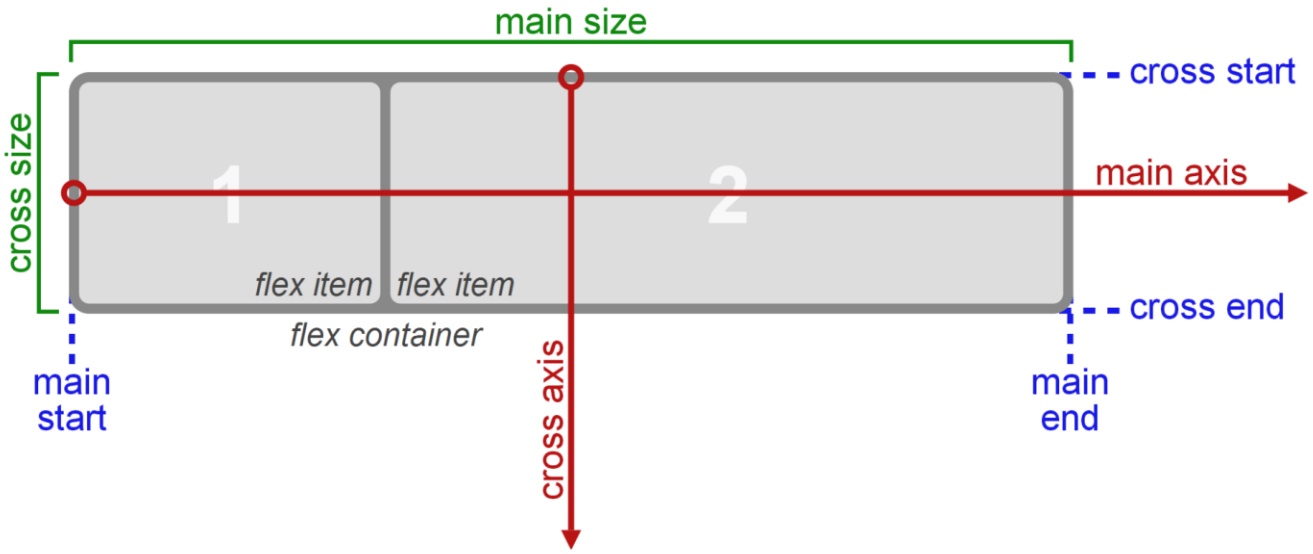
- Элемент при наличии float получает display:block.
  То есть, указав элементу, у которого display:inline свойство float: left/right, мы автоматически сделаем его блочным. В частности, для него будут работать width/height. Есть исключения (см. https://learn.javascript.ru/float)
- Вертикальные отступы margin элементов с float не сливаются с отступами соседей, в отличие от обычных блочных элементов.

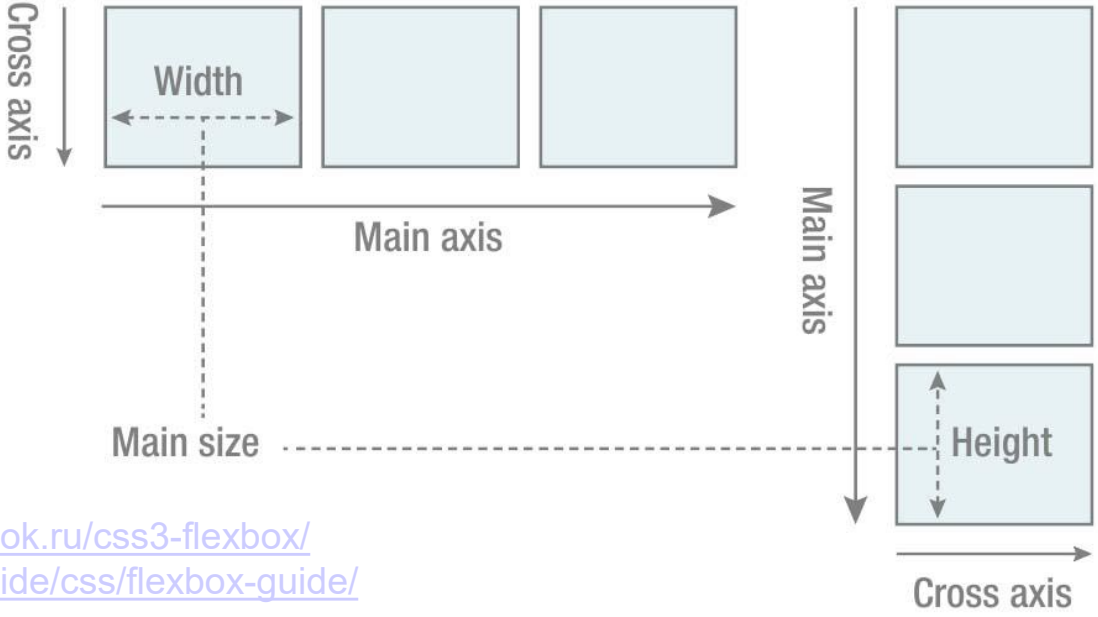
Flexbox

Макет гибкого контейнера — представляет собой способ компоновки элементов, в основе которой лежит идея оси

Flexbox позволяет решать следующие задачи:

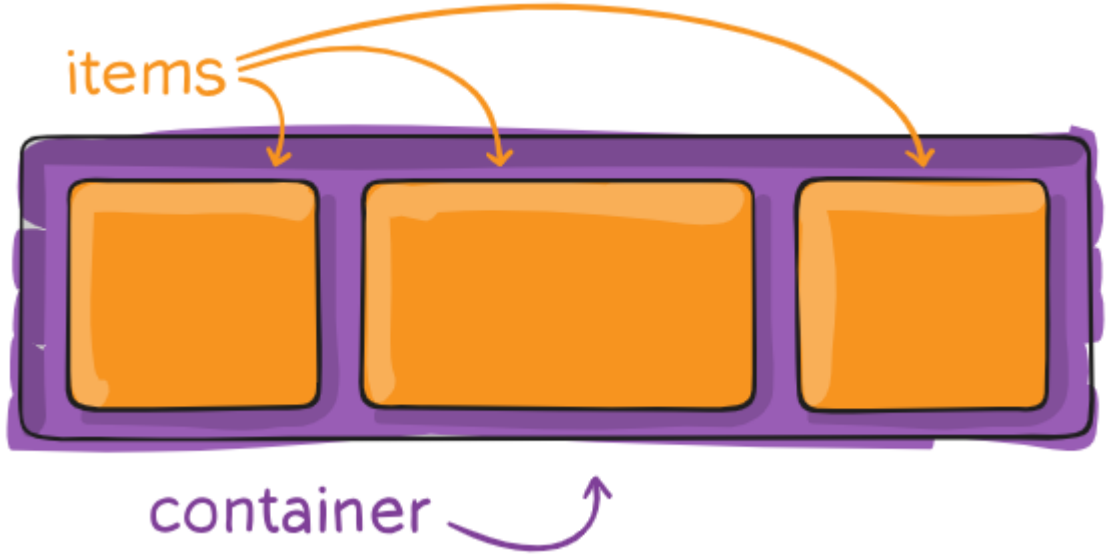
- Располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху вниз или снизу вверх
- Переопределять порядок отображения элементов
- Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство
- Решать проблему с горизонтальным и вертикальным центрированием
- Переносить элементы внутри контейнера, не допуская его переполнения
- Создавать колонки одинаковой высоты
- Создавать прижатый к низу страницы подвал сайта





*- https://html5book.ru/css3-flexbox/
*- https://doka.guide/css/flexbox-guide/

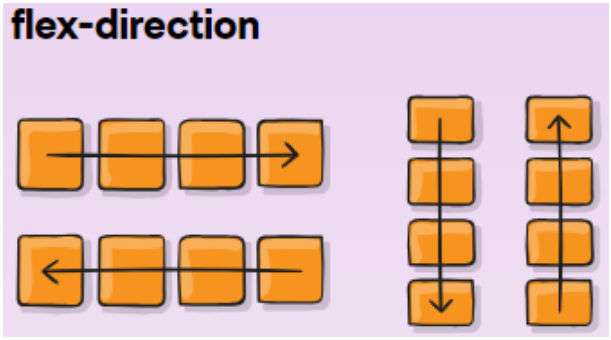
Примеры свойств для контейнера

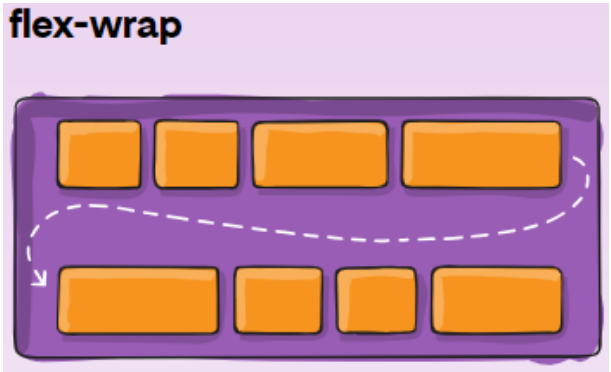


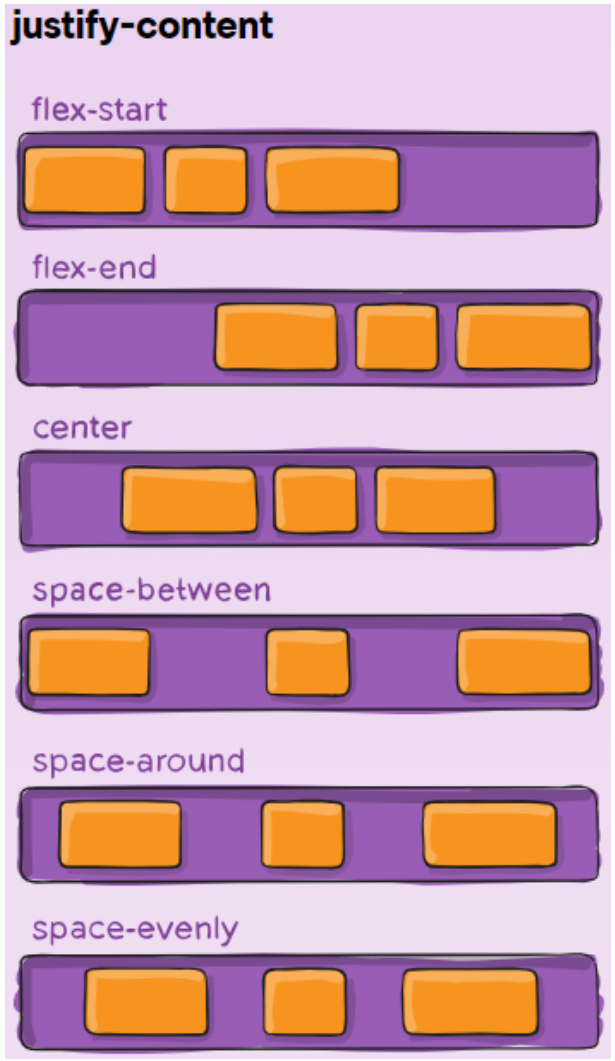

```
.container { display: flex; /* or inline-flex */ }
```

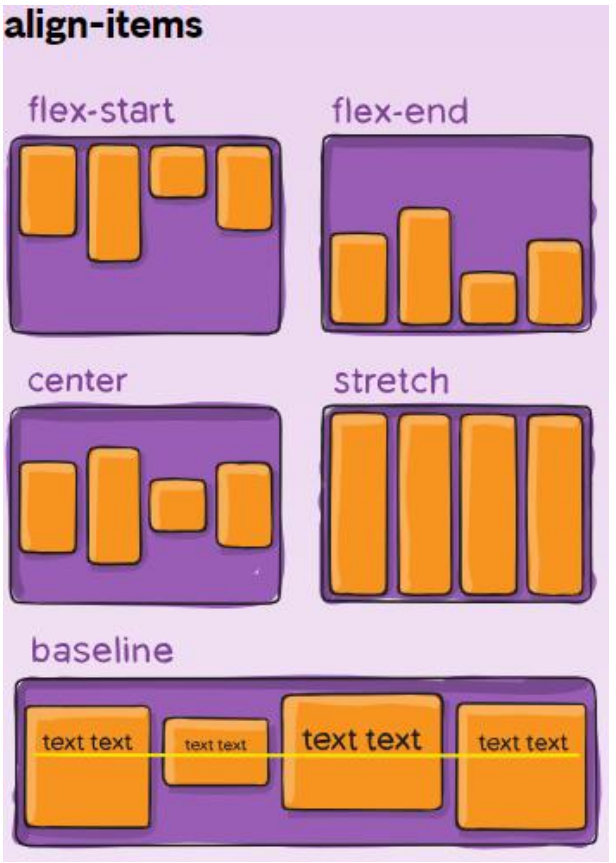
* - https://css-tricks.com/snippets/css/a-guide-to-flexbox

Примеры свойств для контейнера



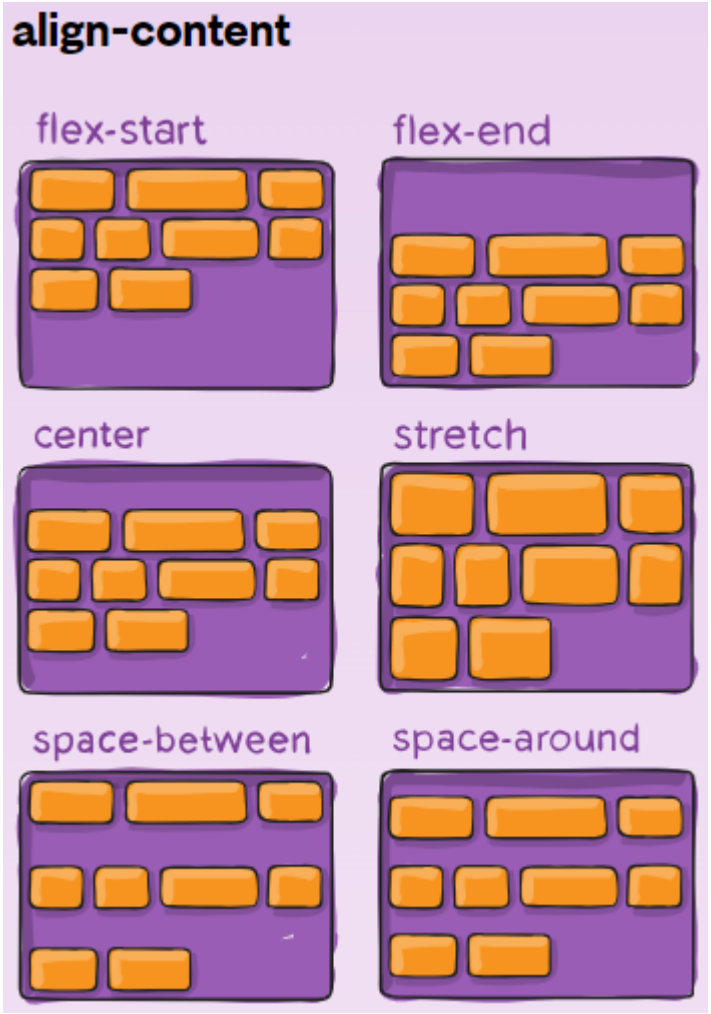






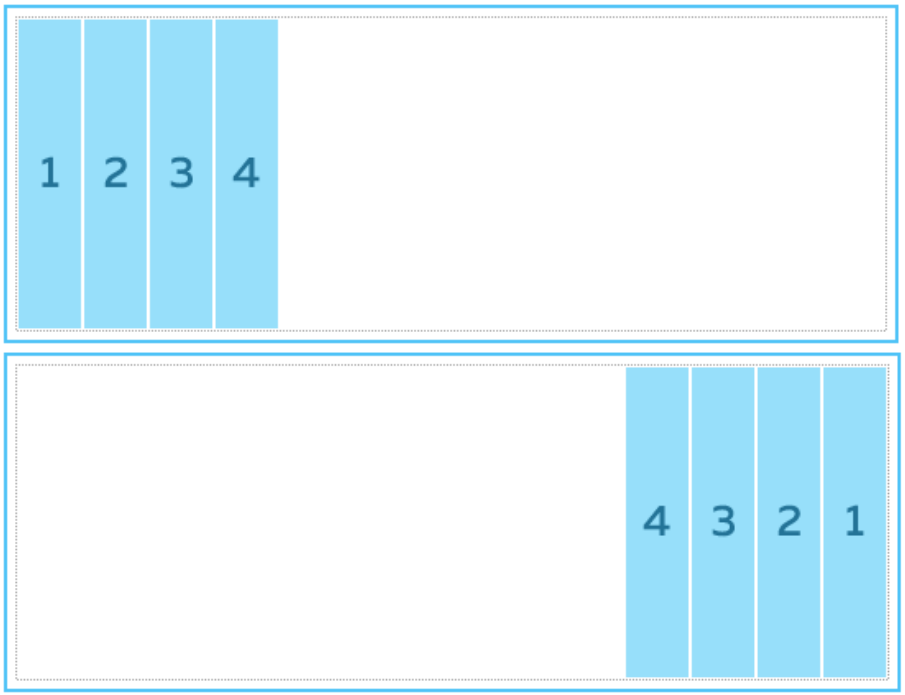
* - https://css-tricks.com/snippets/css/a-guide-to-flexbox

Примеры свойств для контейнера



* - https://css-tricks.com/snippets/css/a-guide-to-flexbox

flex-direction

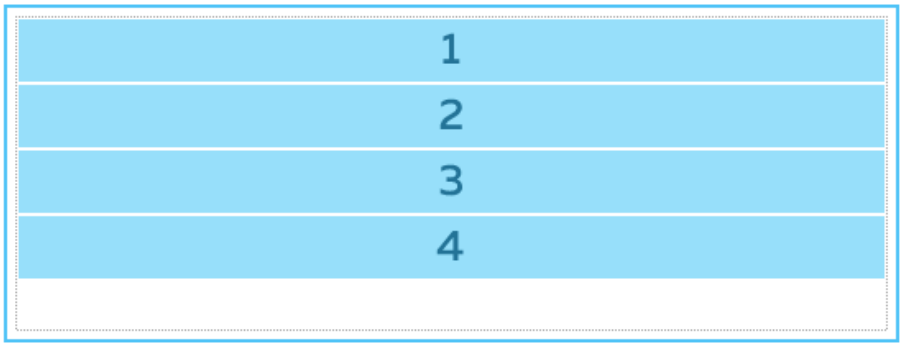



```
.parent {
  display: flex;
  flex-direction: row;
  height: 100%;
}
```

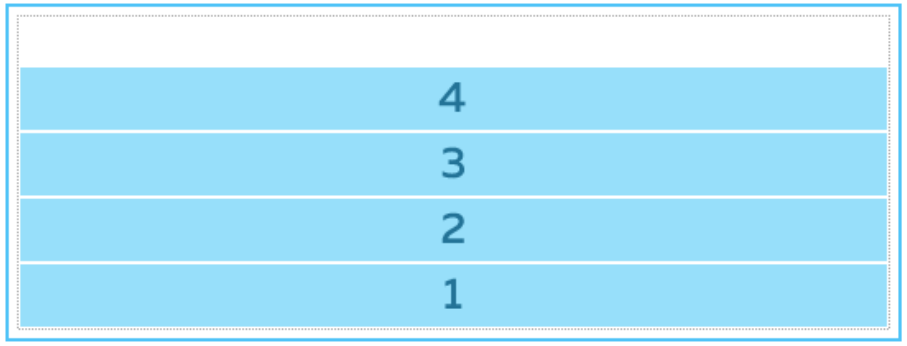
```
.parent {
  display: flex;
  flex-direction: row-reverse;
  height: 100%;
}
```

* - http://tpverstak.ru/flex-cheatsheet/

flex-direction




```
.parent {
  display: flex;
  flex-direction: column;
  height: 100%;
}
```




```
.parent {
  display: flex;
  flex-direction: column-reverse;
  height: 100%;
}
```

* - http://tpverstak.ru/flex-cheatsheet/

Центрирование и растяжение


```css
.top-menu{
    display: flex;
    justify-content: space-around; /*Центрирование*/
}

.top-menu li {
    display: inline-block; /*Для центрирования*/
    padding: 0px 0px;
    margin: 0;
    text-align: center; /*Центрирование*/
    width: 12%; /*Ширина*/
}
```

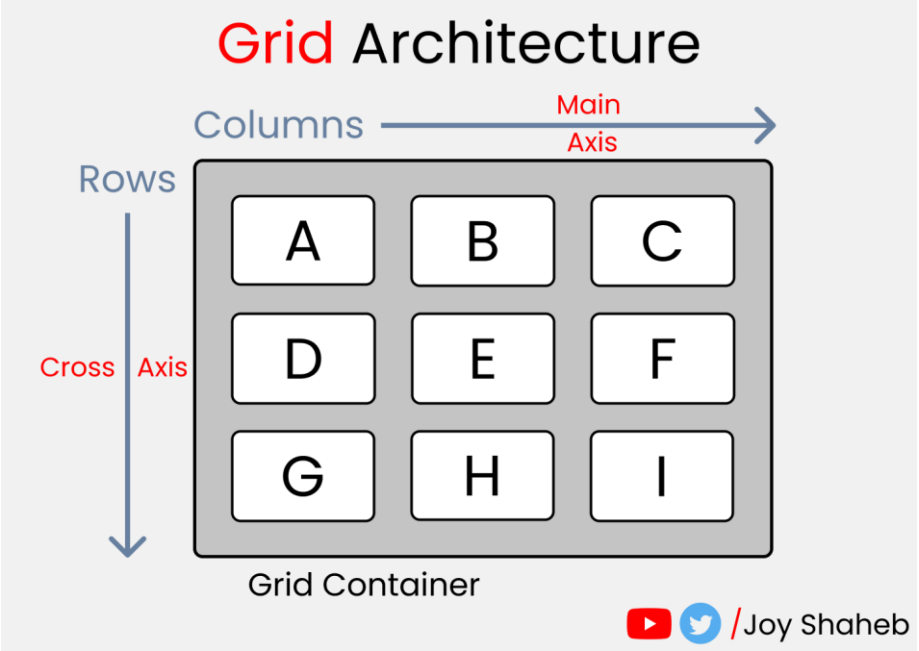
CSS Grid

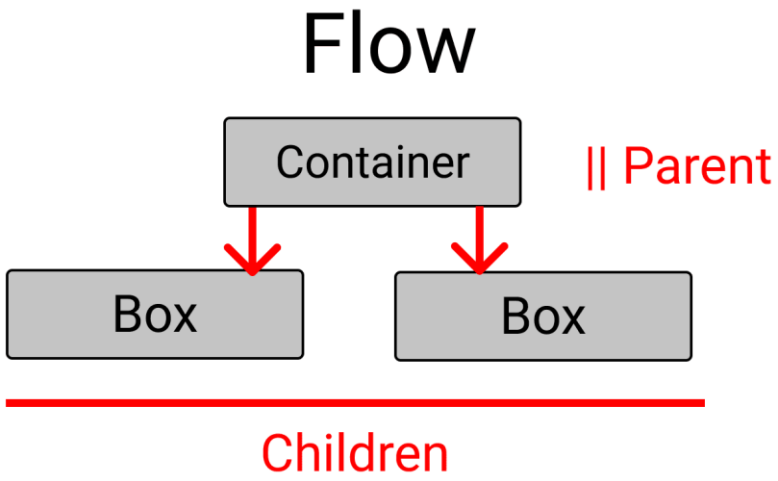
- Grid модуль в CSS был разработан CSS Working Group для того, чтобы предоставить наилучший способ создания шаблонов в CSS. Он попал в Candidate Recommendation в феврале 2017 года, а основные браузеры начали его поддержку в марте 2017 года
- Grid оптимизирована для двумерных шаблонов
- Grid шаблон работает по системе сеток
- Grid это набор пересекающихся горизонтальных и вертикальных линий, которые создают размеры и позиционируют систему координат для контента в grid-контейнере
- 1fr ("одна доля" (fraction) доступного пространства) - элемент займёт одну долю от всего доступного пространства внутри родительского контейнера
- https://css-tricks.com/snippets/css/complete-guide-grid/

Пример

CSS Grid

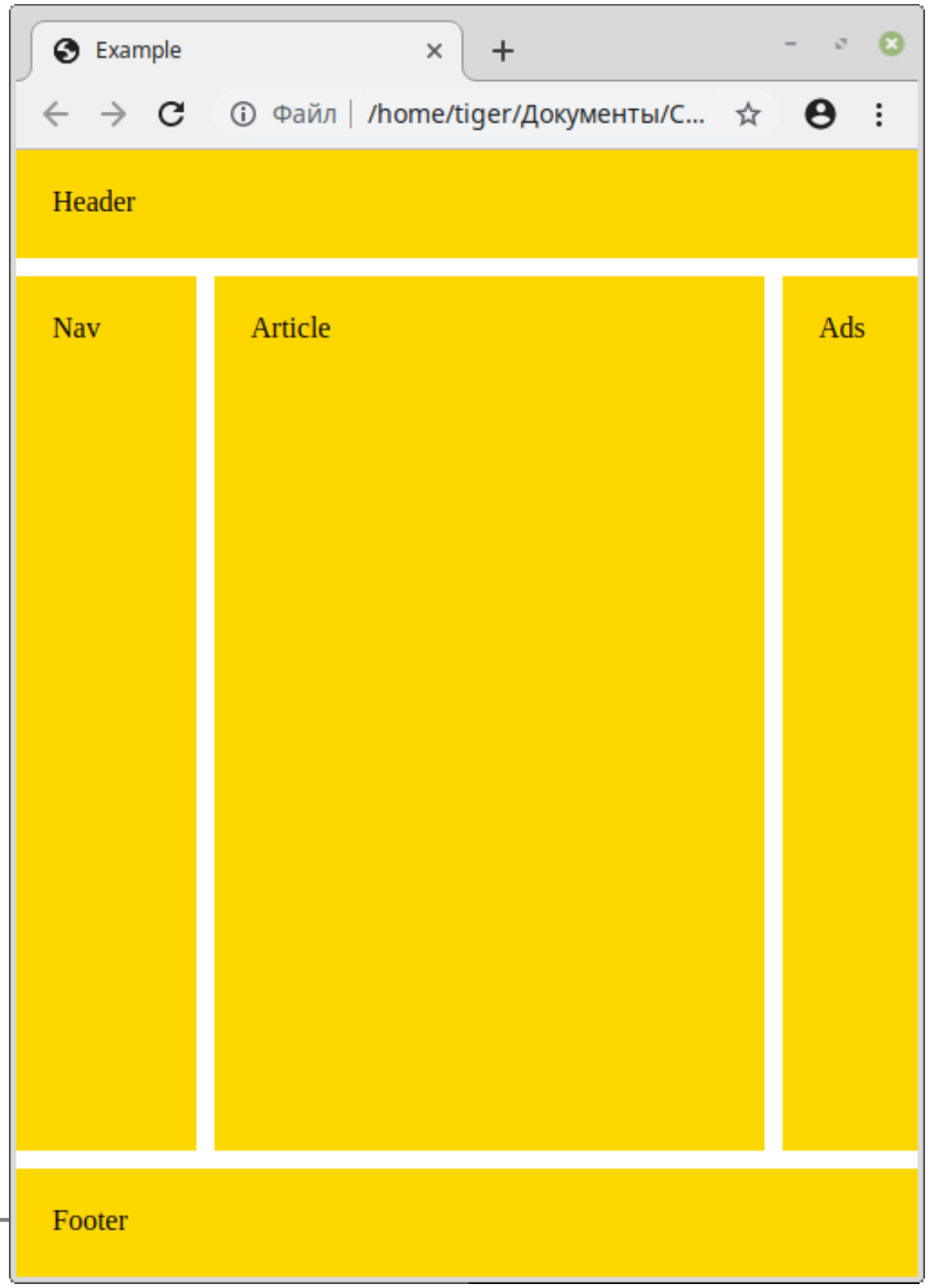
- Грид-модель позволяет позиционировать контент сайта
- Элементы Грида (grid items) располагаются вдоль главной (основной, main) и поперечной (cross) осей (axis)
- При помощи различных свойств можно манипулировать элементами для создания макетов





*- https://habr.com/ru/companies/macloud/articles/564182/
*- https://doka.guide/css/grid-guide/

```html
<!doctype html>
<html>
<head>
<title>Example</title>
<style>
body {
    display: grid;
    grid-template-areas:
      "header header header"
      "nav article ads"
      "footer footer footer";
    grid-template-rows: 60px 1fr 60px;
    grid-template-columns: 20% 1fr 15%;
    grid-gap: 10px;
    height: 100vh;
    margin: 0;}
header, footer, article, nav, div {
        padding: 20px;
        background: gold;}
#pageHeader {grid-area: header;}
#pageFooter {grid-area: footer;}
#mainArticle {grid-area: article;}
#mainNav {grid-area: nav;}
#siteAds {grid-area: ads;}
</style>
</head>
<body>
    <header id="pageHeader">Header</header>
    <article id="mainArticle">Article</article>
    <nav id="mainNav">Nav</nav>
    <div id="siteAds">Ads</div>
    <footer id="pageFooter">Footer</footer>
</body>
</html>
```



Адаптивная верстка

- Адаптивная вёрстка – это такой вид вёрстки, при котором готовый веб-сайт способен подстраиваться под размер и ориентацию любого устройства, а также менять дизайн страницы в зависимости от действий пользователя
- Для каждого отдельного разрешения устройства будет изменяться расположение элементов сайта без необходимости отрисовывать новый дизайн-макет под каждое расширение
- Viewport - это видимая пользователю область веб-страницы, т.е. то, что может увидеть пользователь, не прибегая к прокрутке
  Пример:

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

- Медиа-запросы (media queries) – это правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств

```
@media [тип устройства] [характеристика устройства] {
  /* CSS-правила */
}
```