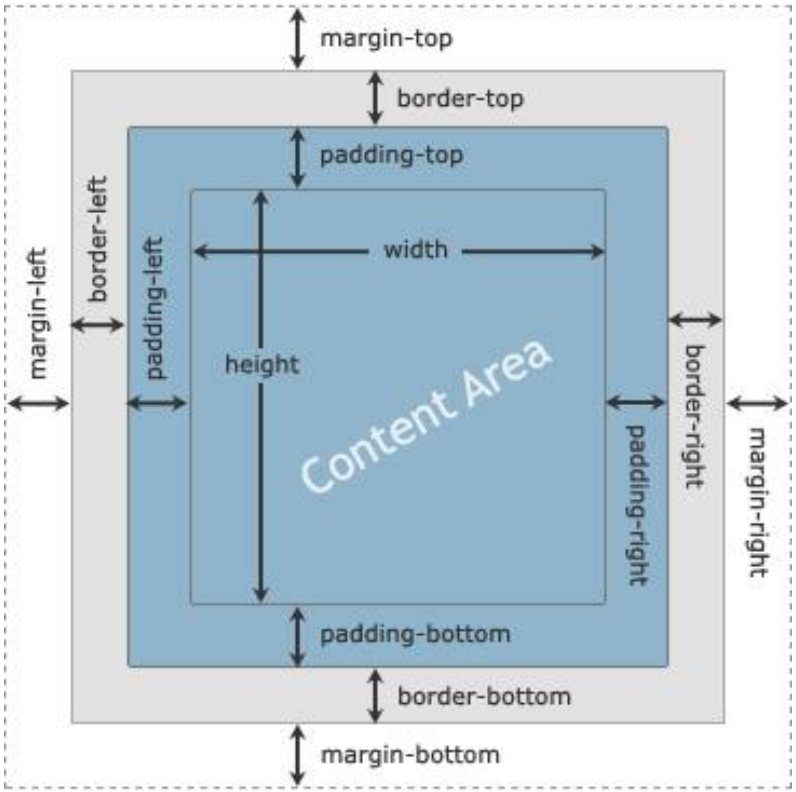
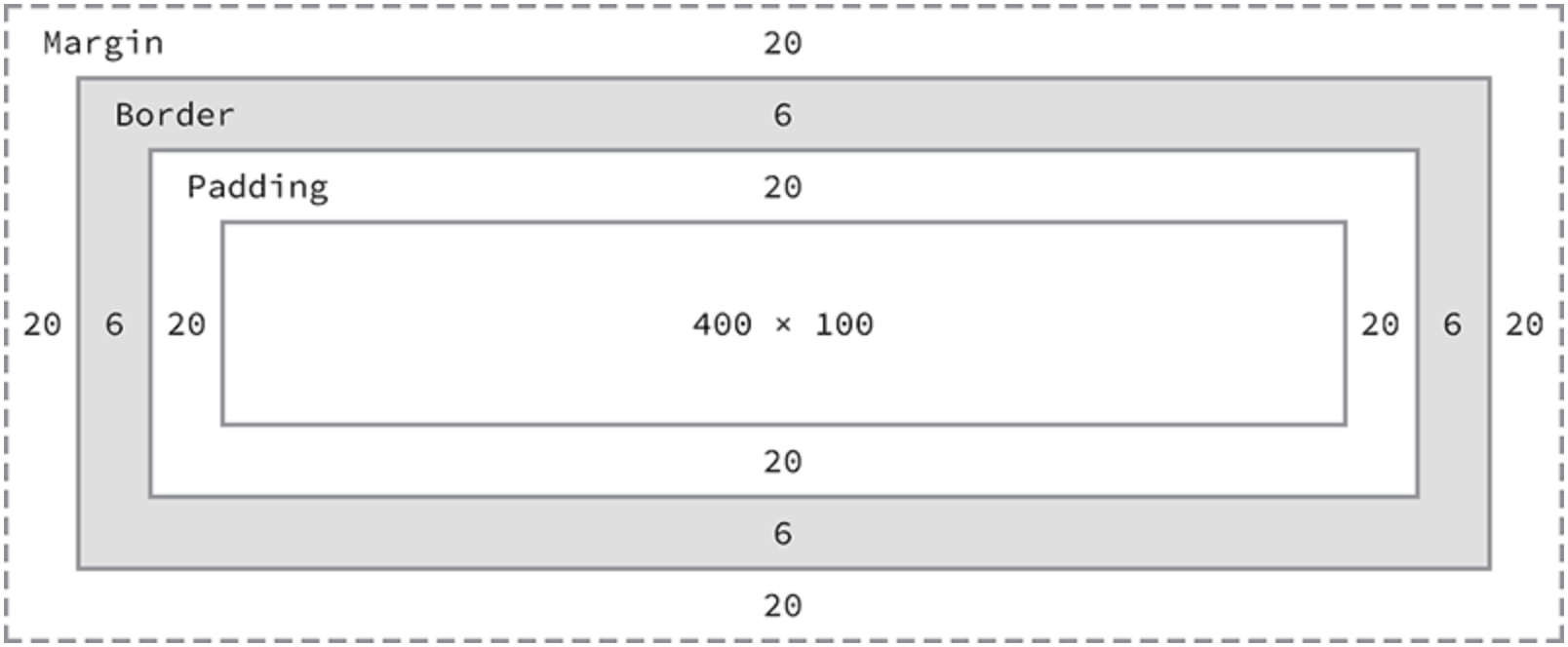
Блочная модель документа
Позиционирование

Блочная модель



http://htmlbook.ru/samlayout/blochnaya-verstka/blochnaya-model

Блочная модель



Ширина: 492px = 20px + 6px + 20px + 400px + 20px + 6px + 20px
Высота: 192px = 20px + 6px + 20px + 100px + 20px + 6px + 20px

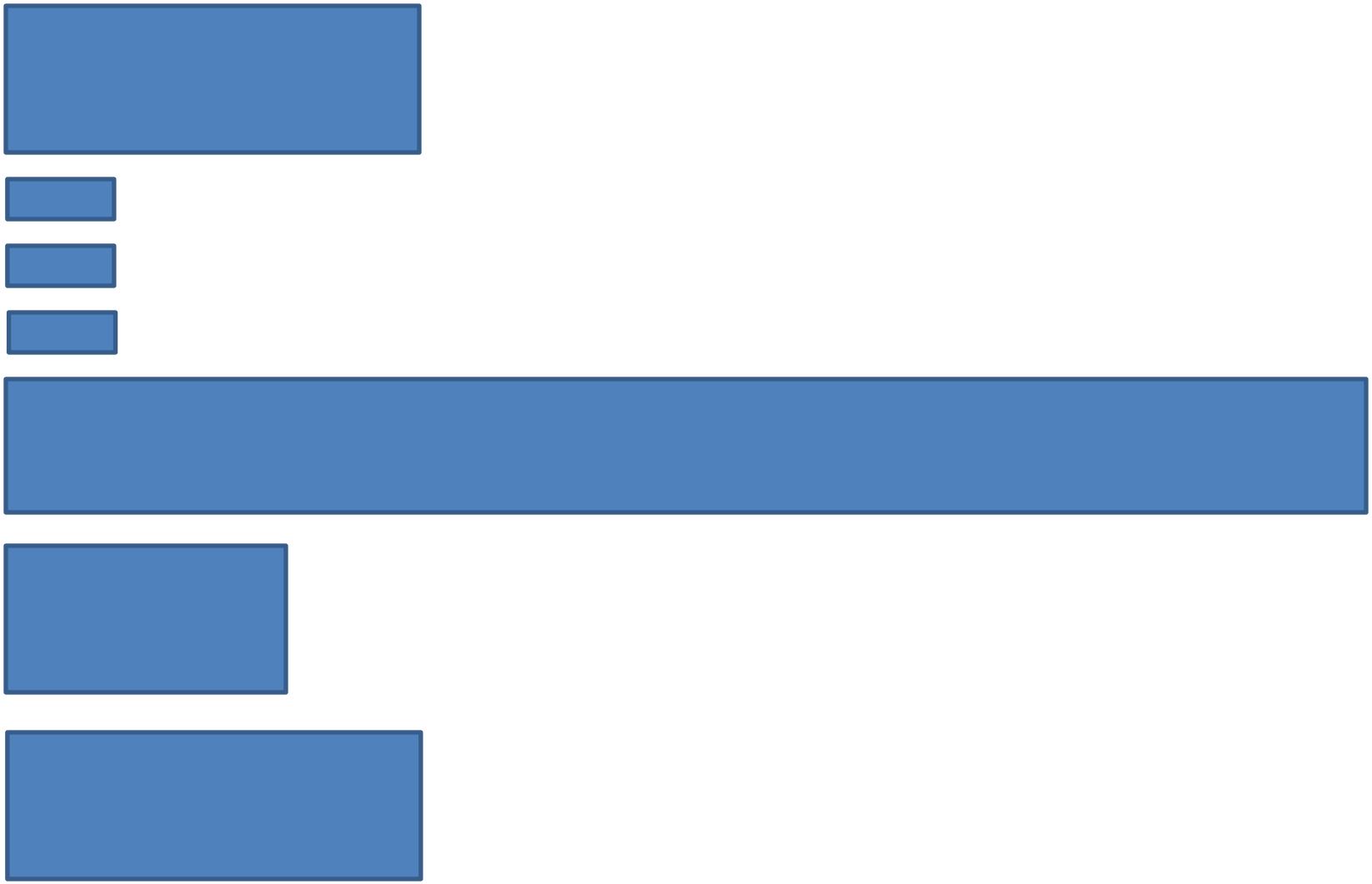
Типы элементов

- Блочные
- Строчные
- Строчно-блочные

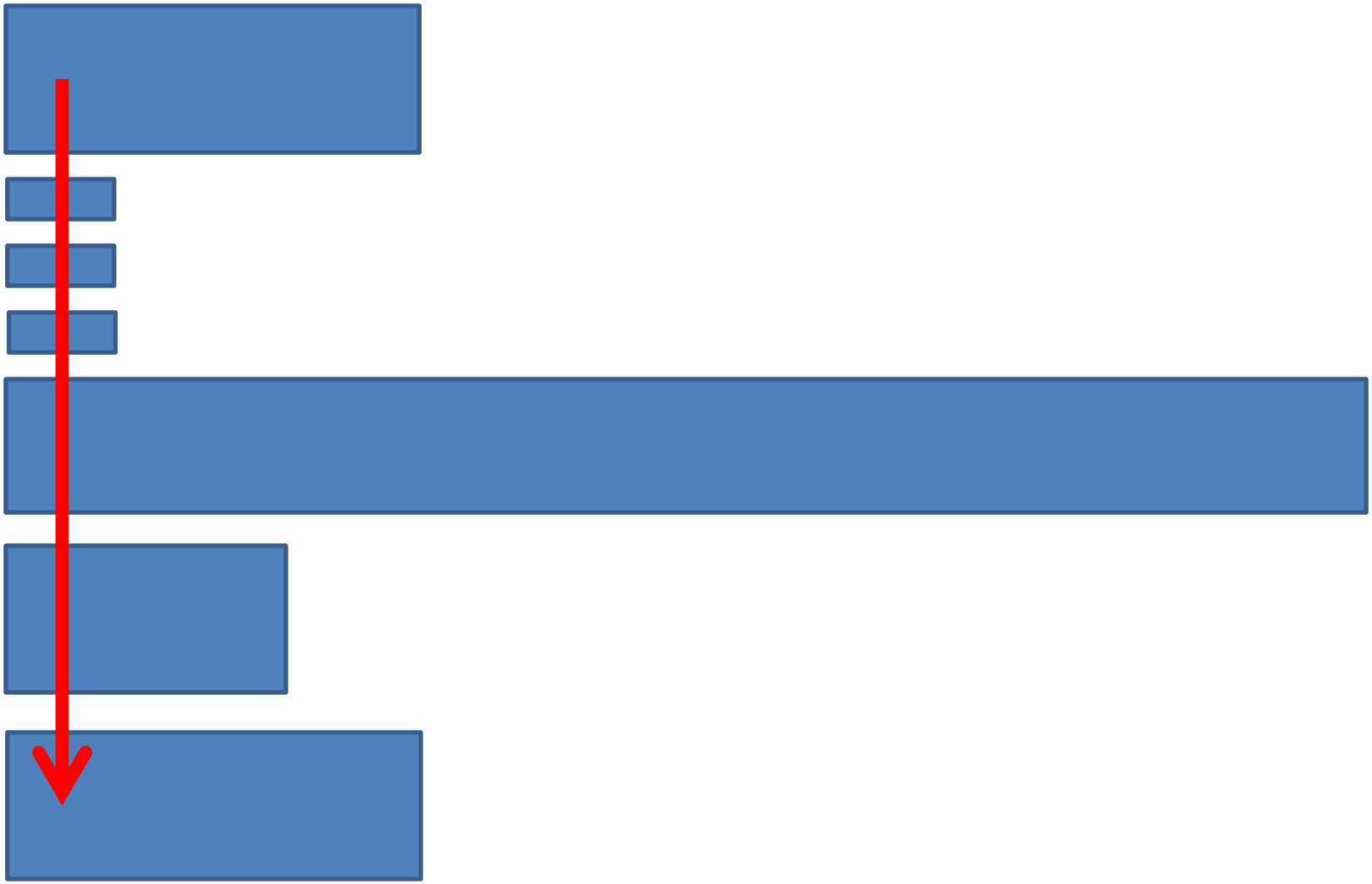
Поведение блочных элементов

- До и после блочного элемента существует перенос строки.
- Блочным элементам можно задавать ширину, высоту, внутренние и внешние отступы.
- Занимают всё доступное пространство по горизонтали.
- Например: ** **, <p>, <h1>, <h2>,

Поведение блочных элементов



Поведение блочных элементов

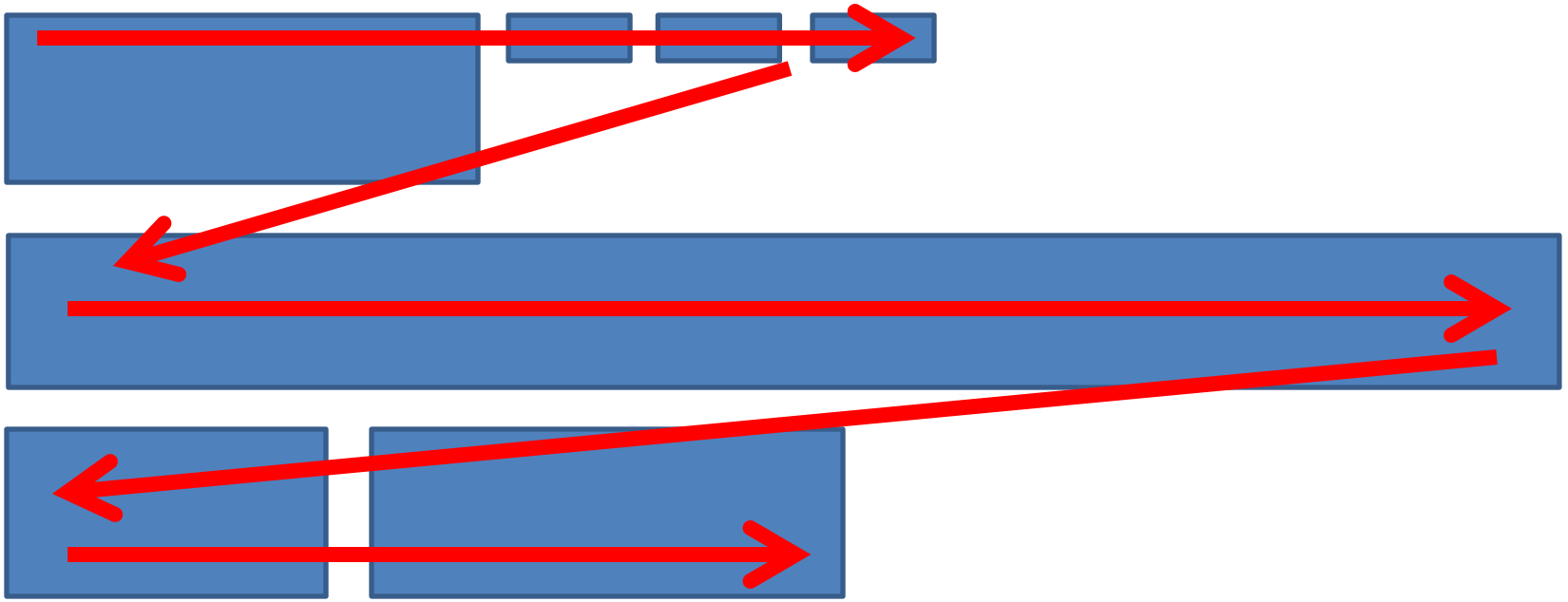


Поведение строчных элементов

- До и после строчного элемента отсутствуют переносы строки.
- Ширина и высота строчного элемента зависит только от его содержания, задать размеры с помощью CSS нельзя.
- Можно задавать только горизонтальные отступы.
- Например: <a>, , ,

Поведение строчных элементов

Поведение строчных элементов



CSS СВОЙСТВА БЛОЧНОЙ МОДЕЛИ

Высота

height: npx | n%

устанавливает высоту блочного элемента, высота не включает толщину границ вокруг элемента, значение отступов и полей

div {height: 150px}

.h {height: 150px} ...

Ширина

width:**n**px | n%

устанавливает ширину блочного элемента, высота не включает толщину границ вокруг элемента, значение отступов и полей

div {width: 100px}

.w {width: 100px} …

Позиция

position: static | absolute | fixed | relative

определяет способ позиционирования блочного элемента относительно окна браузера или других элементов

```
div {position: absolute;  top: 10%; left: 10%}
.pos {position: absolute;  top: 10%; left: 10%}
<div class= pos >...</div>
```

Позиция

- static
  по умолчанию
- relative
  перемещение относительно его **текущей позиции**
- absolute
  перемещение относительно **первого позиционированного предка** (relative, absolute или fixed)
- fixed
  действует как абсолютное, **точкой отсчёта является окно просмотра**

«Координаты»

top, left, right, bottom :npx | n%

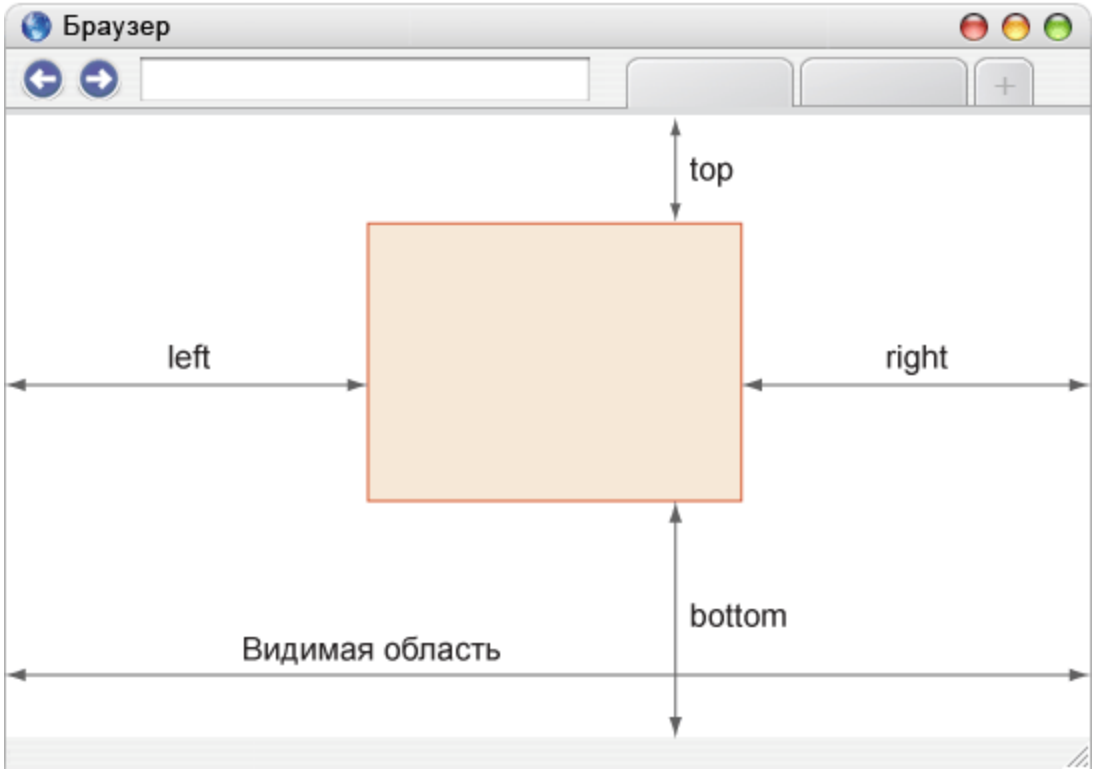
Для **позиционированного элемента** определяет расстояние от левого края родительского элемента, не включая отступ, поле и ширину рамки, до левого края дочернего элемента.

Свойство не работает для элементов у которых postion не задани или задан явно position:static (по-умолчанию)

div {top: 10%; left: 10%}

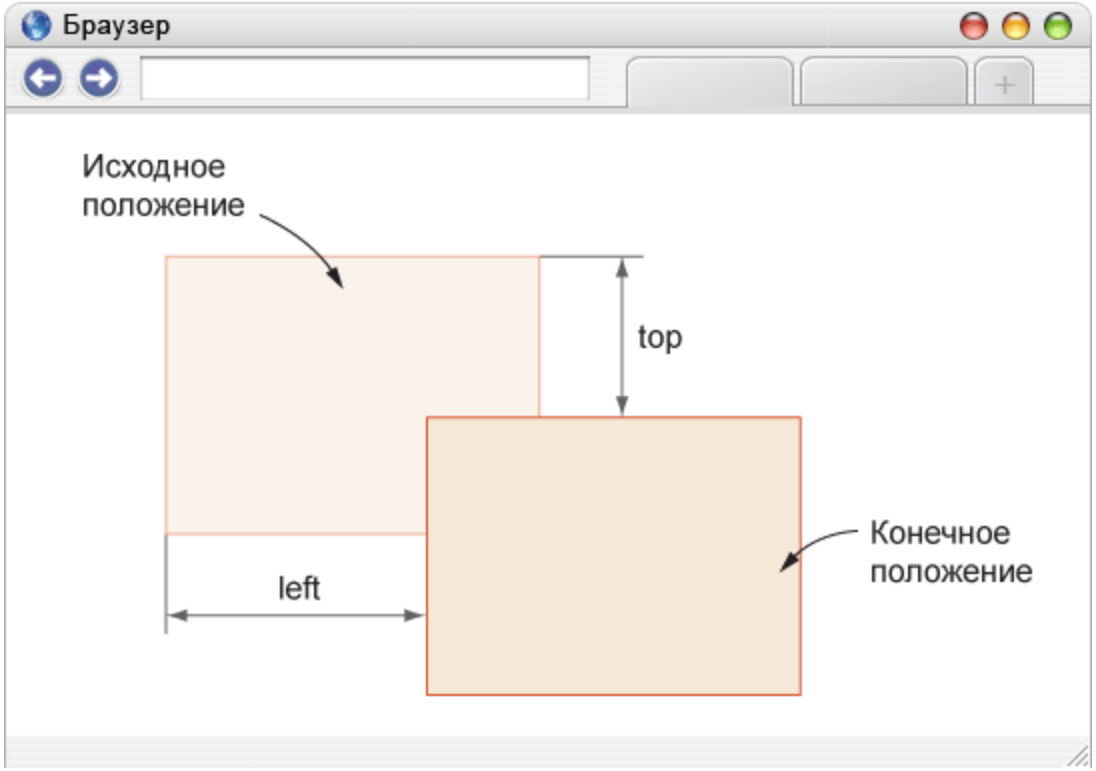
.pos {top: 10%; left: 10%} …

«Координаты» + absolute



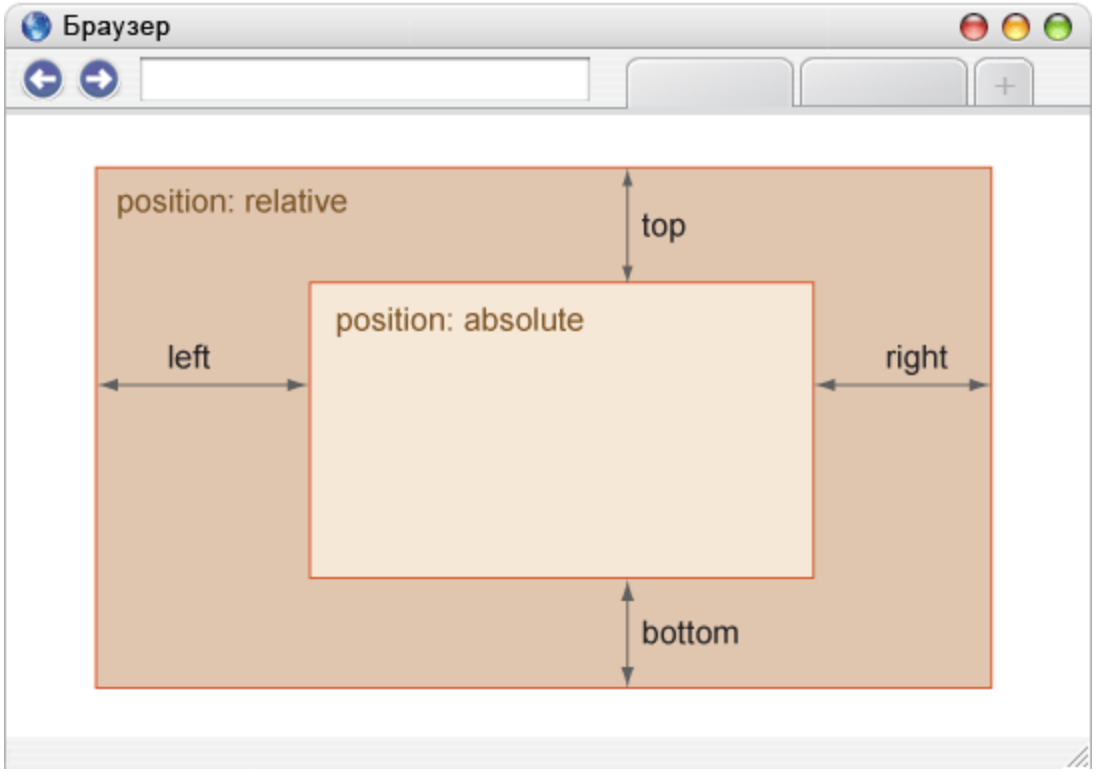
Если *position:absolute*, в качестве родителя выступает окно браузера и положение элемента определяется от его левого края. left + right или left + width

«Координаты» + relative



Если *position:relative*, в качестве родителя выступает исходного положения элемента.

relative + absolute

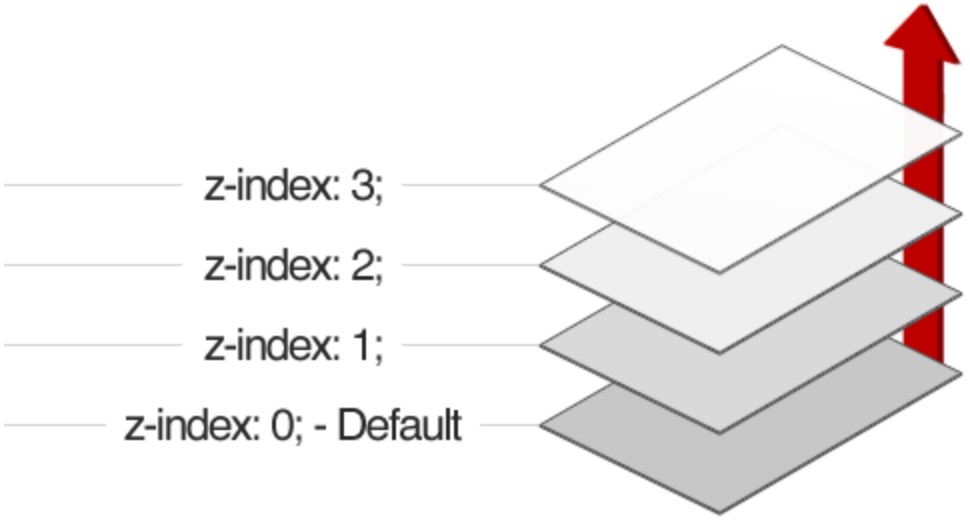


Отсчёт координат ведётся от внутреннего края границы, значения padding не учитываются.

Свойство z-index

z-index:n

Для **позиционированного элемента** определяет положение блочного элемента "в глубину"



ОТСТУПЫ

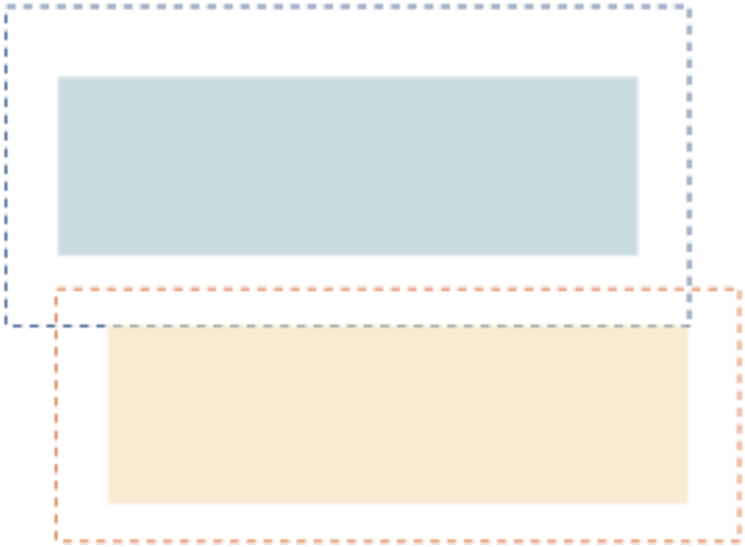
Схлопывание margin

- *Вертикальный* отступ между двумя соседними элементами равен максимальному отступу между ними. Если отступ одного элемента равен 20px, а второго 40px, то отступ между ними будет 40px.

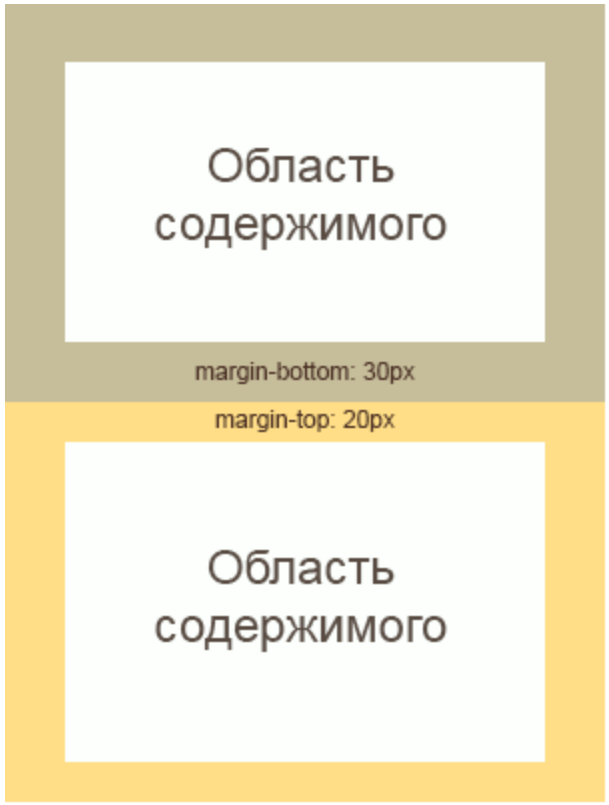
- *Горизонтальные* отступы между элементами просто складываются. Например, горизонтальный отступ между двумя элементами с отступами 30px будет равен 60px.

Схлопывание margin

- Нужно для корректного отображения текста.
- Например расстояние между абзацами (тег <p>) без схлопывания увеличится в два раза



Схлопывание margin

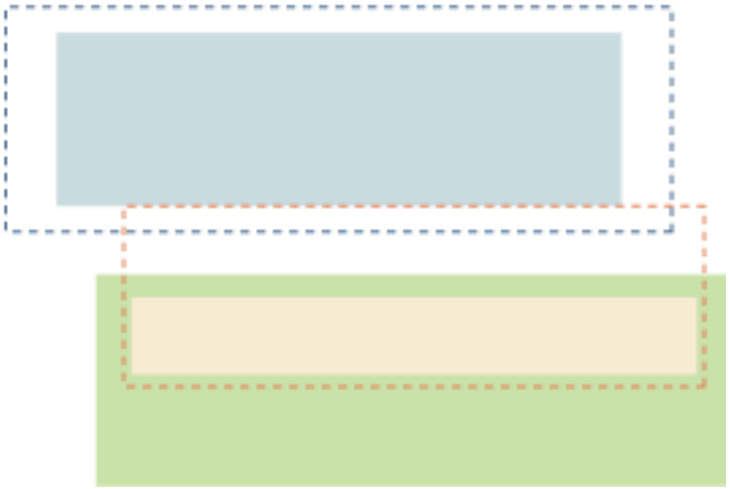


Выпадание margin

- Если внутри родительского блока расположить блок и задать ему отступ сверху, то внутренний блок прижмется к верхнему краю родительского, а у родительского элемента появится отступ сверху. Т.е. верхний отступ внутреннего элемента «выпадает» из родительского элемента.
- Если у родительского элемента тоже был задан внешний отступ, то выберется максимальный отступ между собственным и «выпавшим» .

Выпадание margin

Предположим, что в нижнем блоке располагается дочерний элемент, у которого задан верхний отступ. Из блочной модели следует, что такой отступ сдвигает дочерний элемент вниз относительно верхнего края родителя. Однако с учётом схлопывающихся отступов результат будет иным



Отмена схлопывания

- для элементов установлено свойство padding.
- для элементов на стороне схлопывания задана граница;
- на элементах с абсолютным позиционированием, т.е. таких, у которых position установлено как absolute;
- на плавающих элементах (для них свойство float задано как left или right);
- для строчных элементов;
- для **<html>**

Центрирование

Чтобы отцентровать *блочный* элемент, нужно выполнить следующие действия:

- Задать элементу ширину, которая меньше ширины родительского контейнера.
- Задать для внешних отступов справа и слева значение auto.

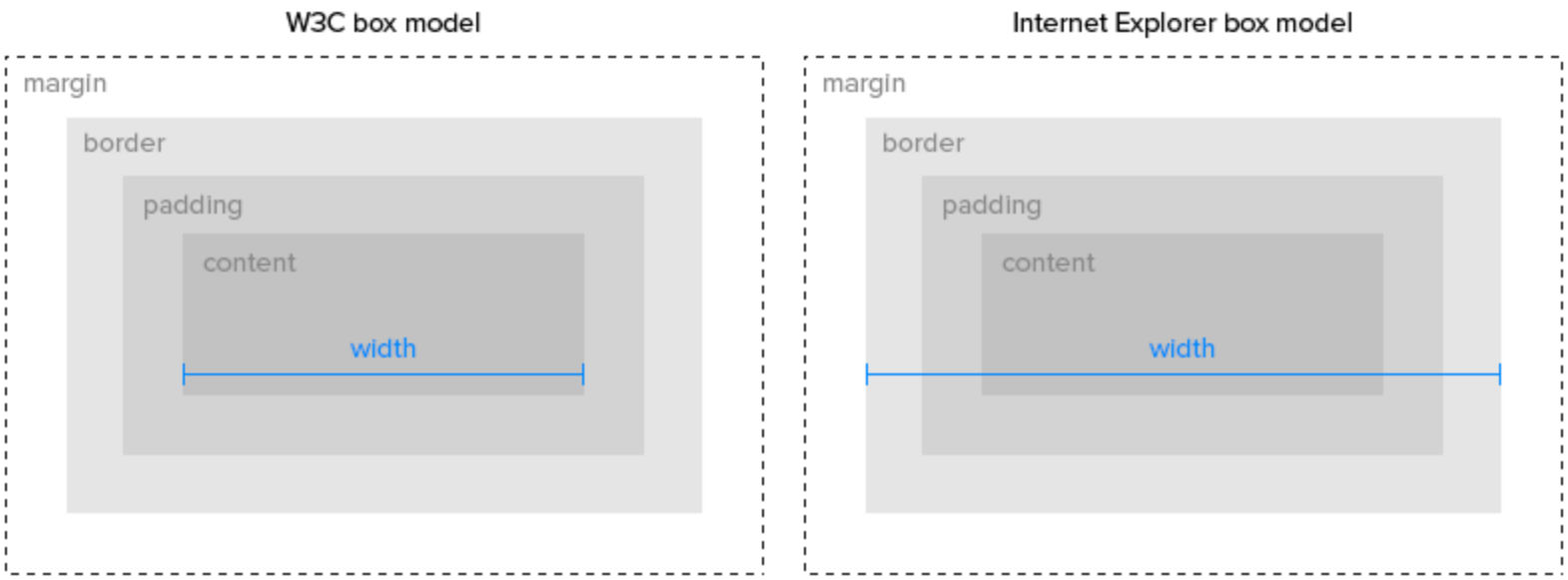
- Или просто использовать FlexBox

Проблема блочной модели

```
<div class="block" style="width: 100px;">
    <input type="text" value="Фамилия">
</div>
 <div class="block" style="width: 150px;">
    <input type="text" value="Имя">
 </div>
<div class="block" style="width: 200px;">
     <input type="text" value="Отчество">
</div>
```

```
.block {
      padding: 10px;
}
input[type="text"] {
      width: 100%;
      padding: 5px 15px;
}
```

CSS свойство box-sizing



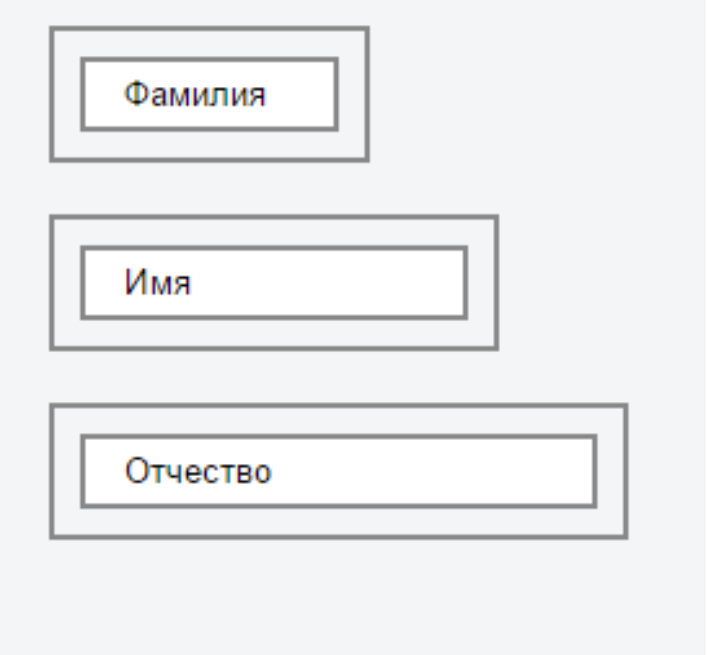
CSS свойство box-sizing

Применяется для изменения алгоритма расчёта ширины и высоты элемента.

- **content-box** - width и height задают ширину и высоту контента и не включают в себя значения отступов, полей и границ.
- **border-box** width и height включают в себя значения полей (padding) и границ (border), но не отступов (margin).

CSS свойство box-sizing

```
input[type="text"] {
  ...
  box-sizing: border-box;
}
```



CSS свойство display

Наиболее распространённые

- block
- inline
- inline-block
- none

CSS свойство display

```
a {
display: block;
}
/*превращаем строчный элемент в блочный, благодаря чему весь блок, а не только текст ссылки, становится ссылкой*/
div {
display: inline;
}
 /*превращаем блочный элемент в строчный*/
```

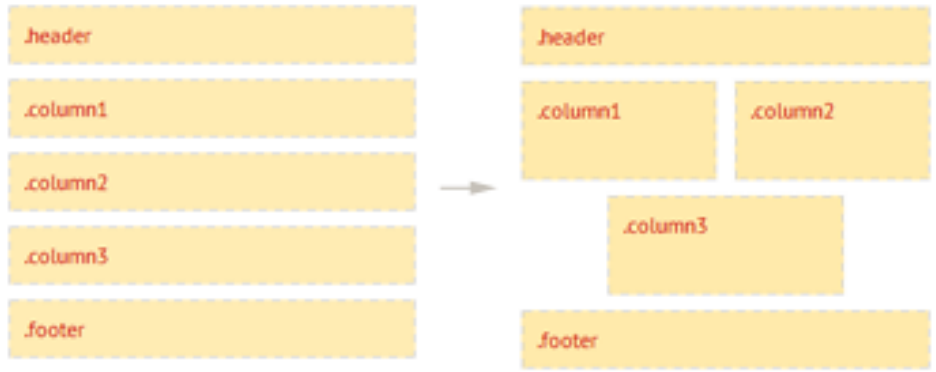
http://frontender.info/adaptive-vs-responsive-terminology/

ПОТОК ДОКУМЕНТА

Поток документа

Исходный код

```
<div class="header"></div>
<div class="column1"></div>
<div class="column2"></div>
<div class="column3"></div>
<div class="footer"></div>
```

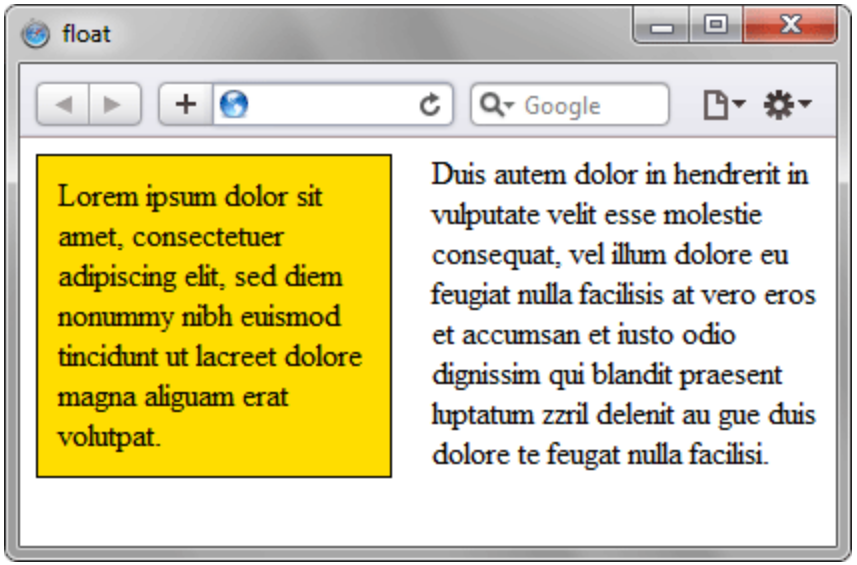


Обтекание

- Свойство float берёт элемент, убирает его из обычного потока документа и позиционирует слева или справа от родительского элемента. Все остальные элементы на странице будут обтекать такой элемент.
- float: left | right

CSS свойство float

```
<div class=layer1>
Lorem ....
</div>
<div>
Duis ...
</div>
```



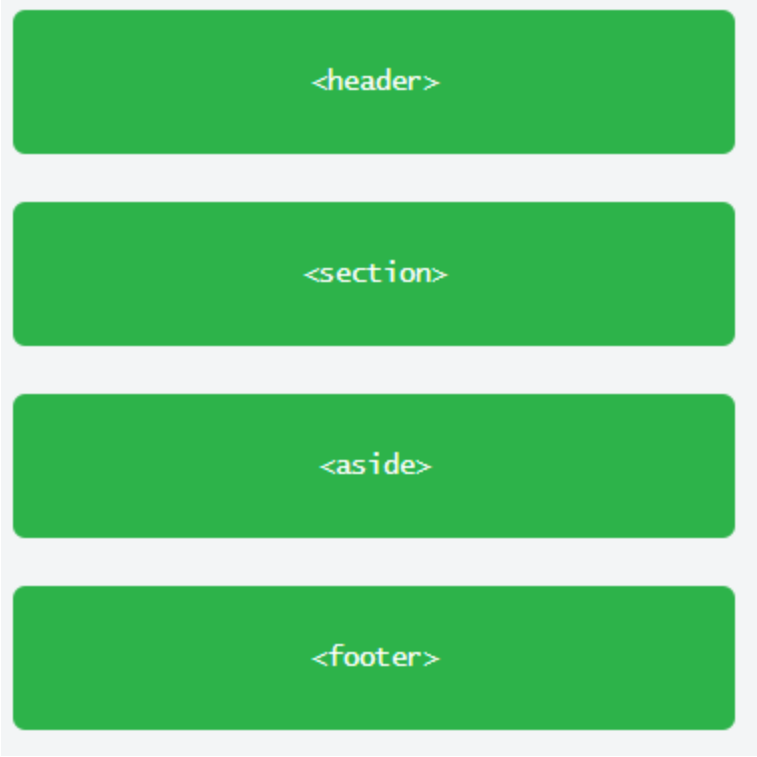

```
.layer1 {
    float: left;
}
```

CSS свойство clear

- Назначение: возвращение нормального потока документа
- Очистка float происходит с помощью свойства clear, которое принимает несколько различных значений: наиболее часто используемые значения — left, right и both.

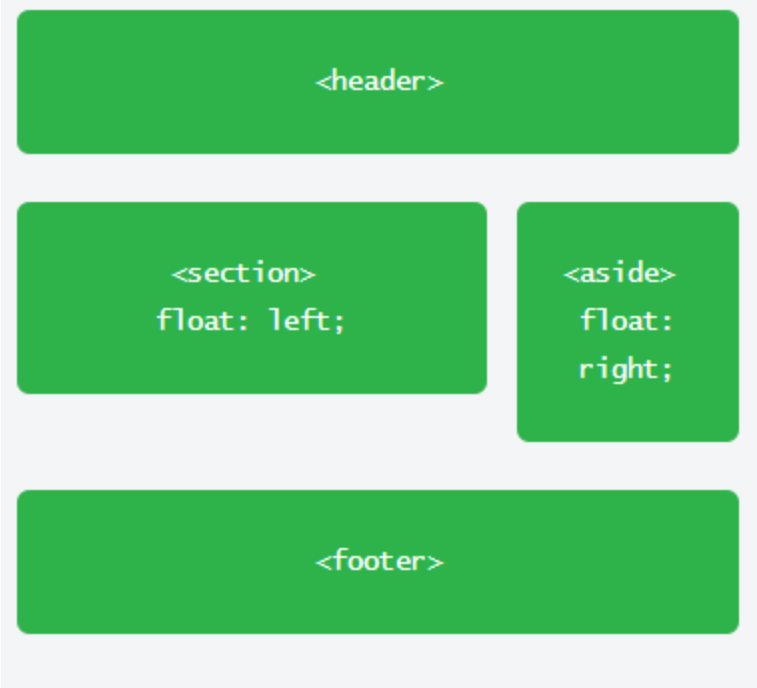
Позиционирование

- <header>header </header>
- <section>section </section>
- <aside> aside </aside>
- <footer> footer </footer>



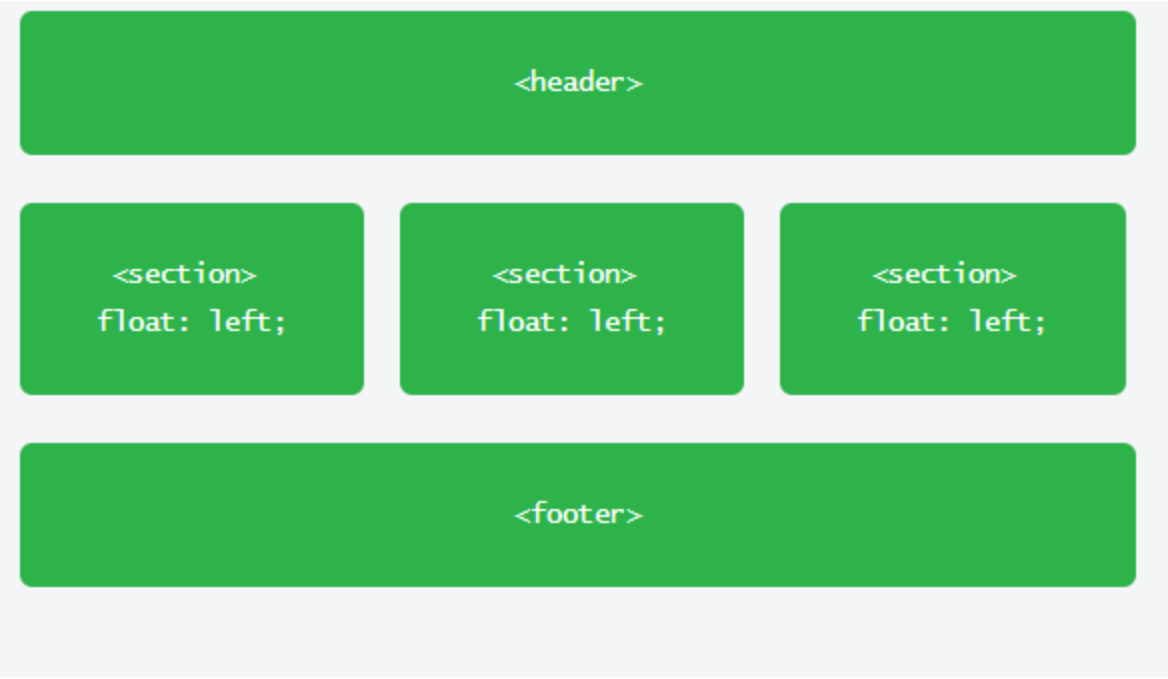
Позиционирование

```
section {
float: left;
width: 63%; }
aside {
float: right;
width: 30%; }
footer {
clear: both;
margin-bottom: 0; }
```



Позиционирование

```
section {
float: left;
width: 30%;
}
footer {
clear: both;
margin-bottom: 0;
}
```



https://webref.ru/layout/learn-html-css/positioning-content

Display: inline-block

Строчные черты:

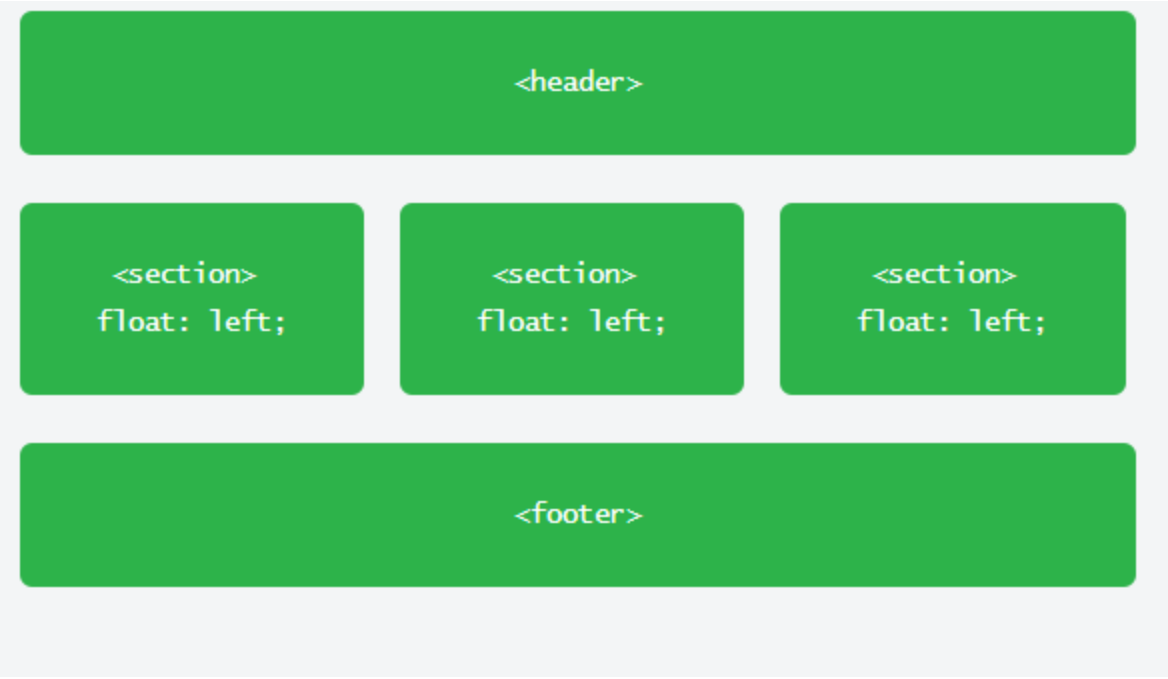
- по ширине они ужимаются под своё содержимое;
- могут располагаться в одну строку;
- реагируют на вертикальное выравнивание, vertical-align;
- реагируют на горизонтальное выравнивание, text-align, заданное у родителя.

Блочные черты:

- можно задавать размеры с помощью width и height;
- внешние и внутренние отступы и рамки, которые работают во всех направлениях и увеличивают размер элемента.

Display: inline-block

```
section {
float: left;
width: 30%;
}
footer {
clear: both;
margin-bottom: 0;
}
```



https://webref.ru/layout/learn-html-css/positioning-content

float против inline-block

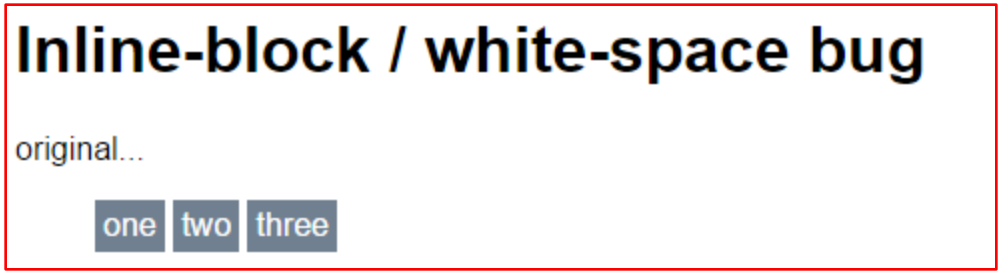
- float

- inline-block

inline-block и пробелы в коде

```
<h1>white-space bug
</h1>original... <ul>
<li>one</li>
<li>two</li>
<li>three</li>
</ul>
```

```
ul { list-style: none }
li {display: inline-block; }
```



- http://dabblet.com/gist/2422174
- https://htmlacademy.ru/blog/21-fighting-the-space-between-inline-block-elements

http://frontender.info/adaptive-vs-responsive-terminology/

http://www.liquidapsive.com/

ВЁРСТКА

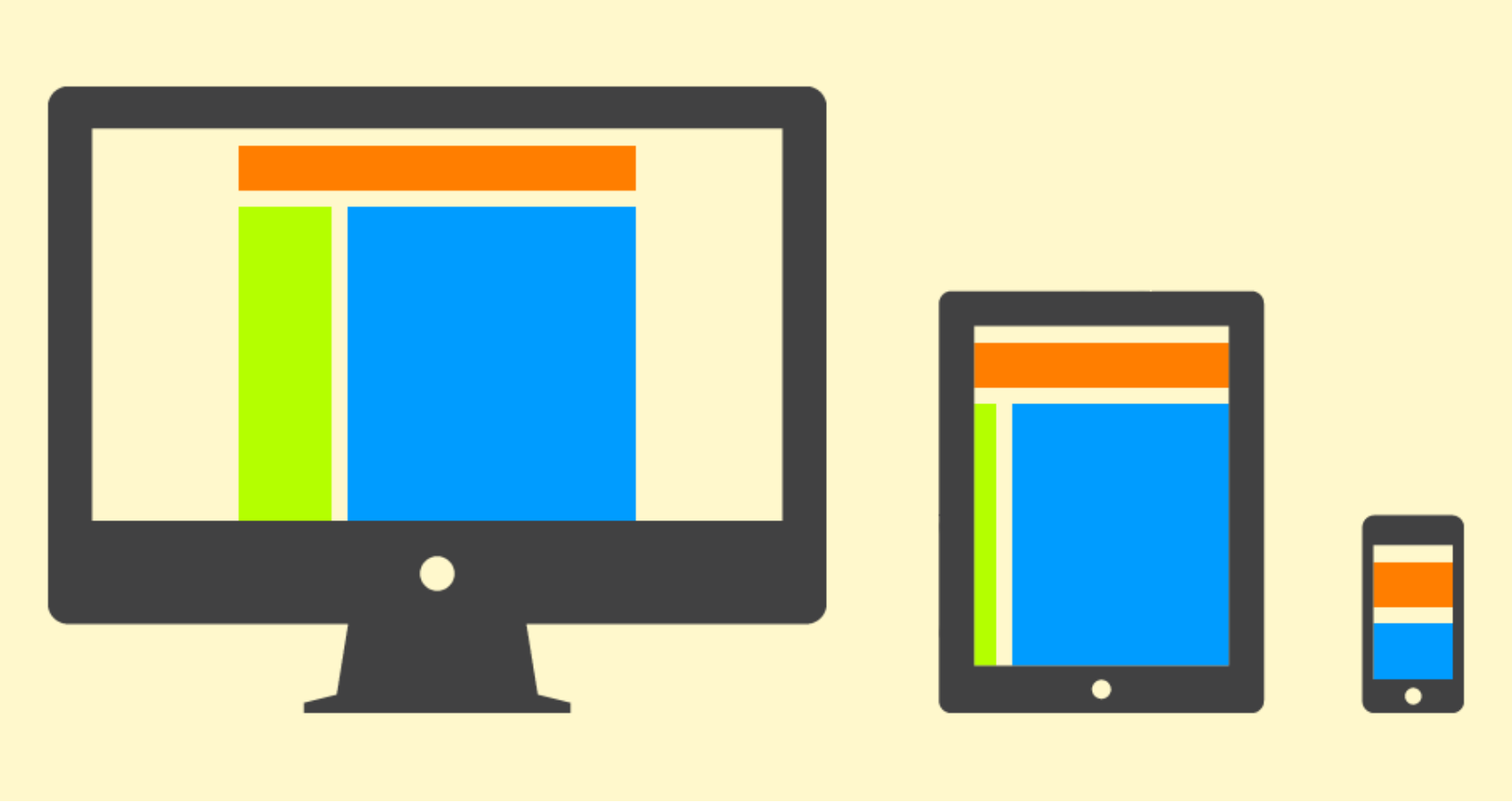
Верстка

- Фиксированная (px)
- Резиновая (%)
- Адаптивная (px + media-queries)
- Отзывчивая (%+ media-queries)

Фиксированная верстка (static)

- ***px***
- Макет страницы, ширина контента которой жестко задана в пикселях и не меняется в зависимости от размеров окна браузера.
- **<style>**
  .static { width:440px }
  </style>

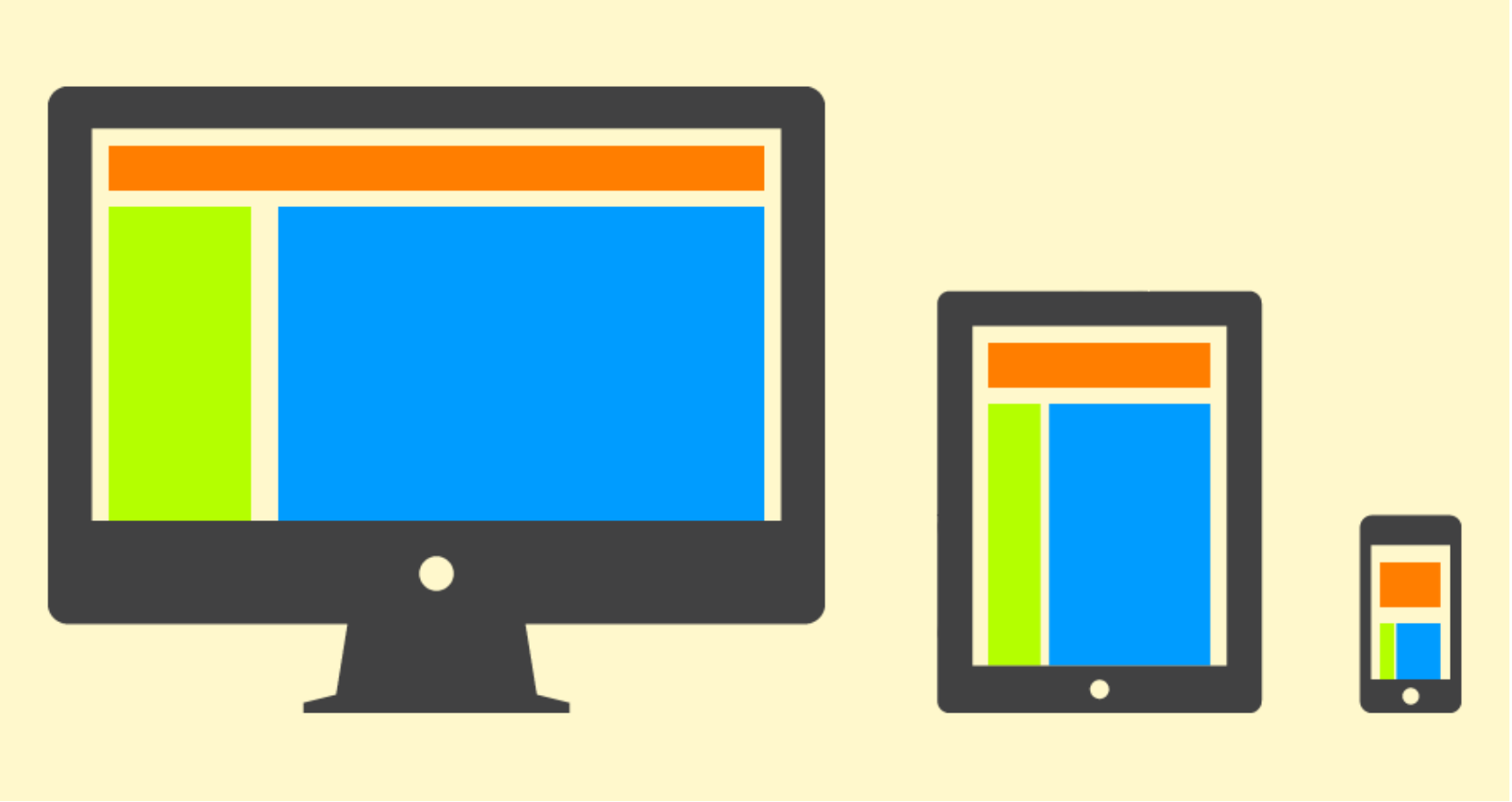
Фиксированная верстка



Резиновая верстка (liquid)

- **%**
- Макет страницы, где содержимое принимает размер любого экрана за счет использования для структурных элементов относительных показателей вместо статических.
- **<style>**
  .liquid { width:50% }
  </style>

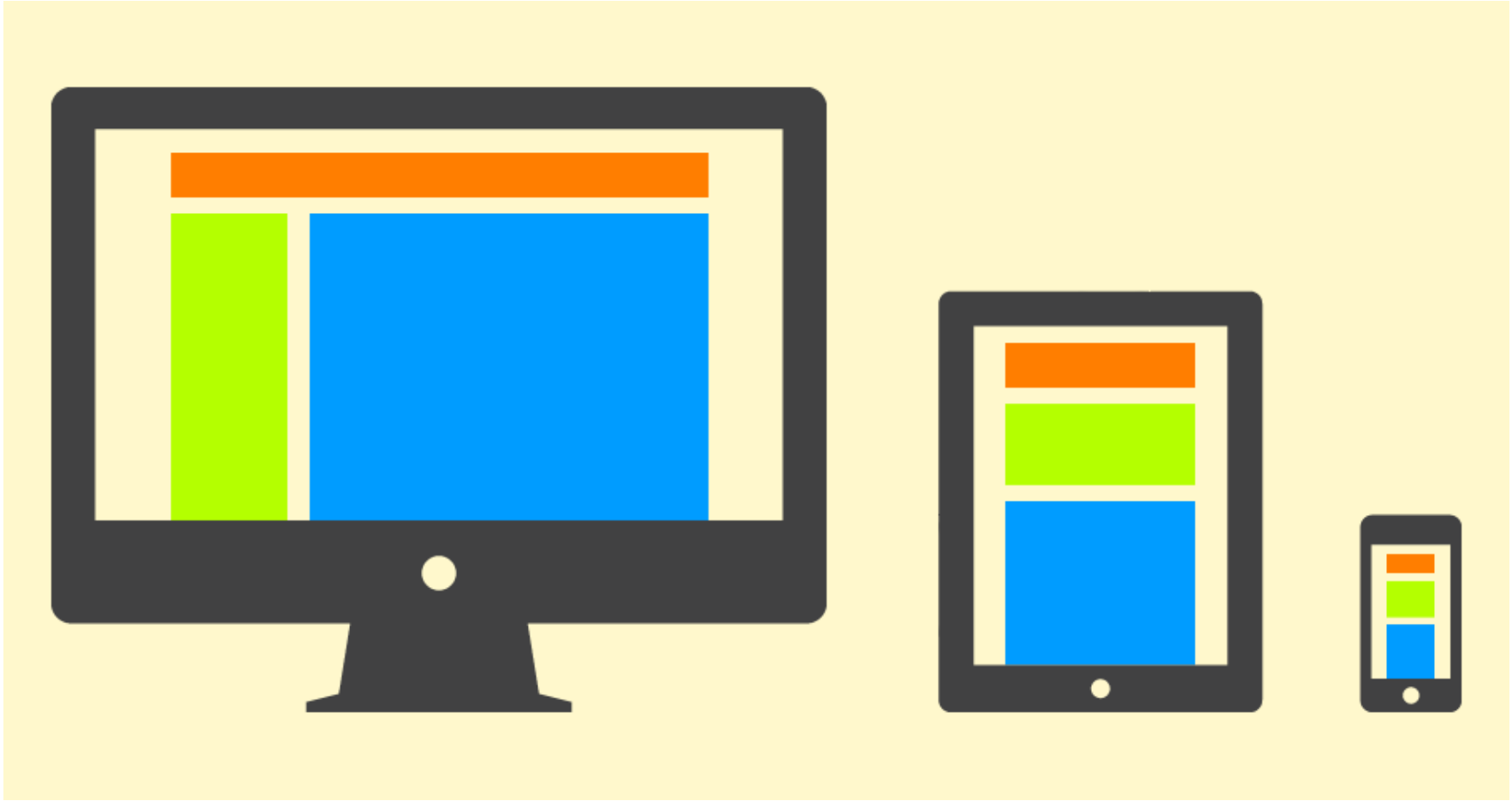
Резиновая верстка



Адаптивная верстка (adaptive)

- ***px + media-queries***
- Макет страницы, состоящий из набора фиксированных макетов для работы с различными разрешениями экранов
- **<style>**
  . adaptive { width: 430px }
  @media (max-width: 680px) {
  .adaptive { width: 200px } }
  </style>

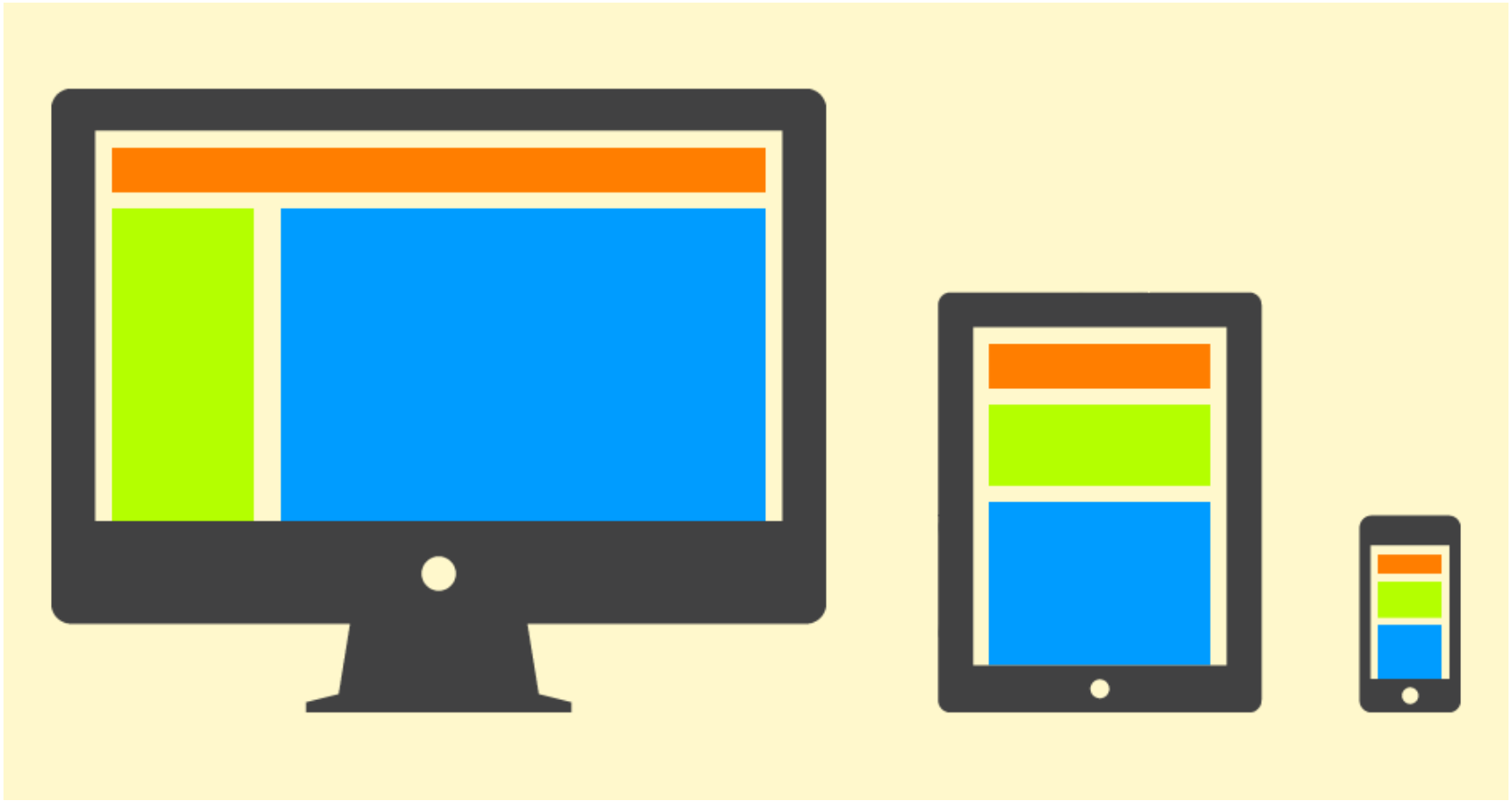
Адаптивная верстка



Отзывчивая верстка (liquid)

- ***% + media-queries***
- Комбинация адаптивной и резиновой верстки
- **<style>**
  .responsive { width: 50% }
  @media (max-width: 960px){
  .responsive { width: 100% } }
  </style>

Отзывчивая верстка



ПОДХОДЫ К ВЕРСТКЕ

Способы верстки

- На основе <table> (Устаревший подход)
- На основе html5 и тегов

```
<body>
  <div id="container">
      <header></header>
      <nav></nav>
      <article></article>
      <footer></footer>
  </div>
</body>
```

```
<body>
  <div id="container">
    <div id="header"></div>
    <div id="sidebar"></div>
    <div id="content"></div>
    <div id="footer"></div>
  </div>
</body>
```

HTML5 теги

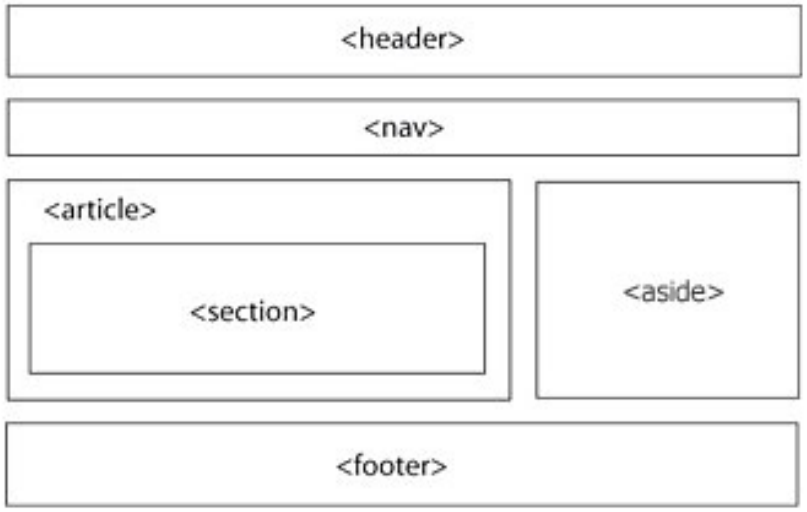


Fig 3. Semantic HTML, using structural elements

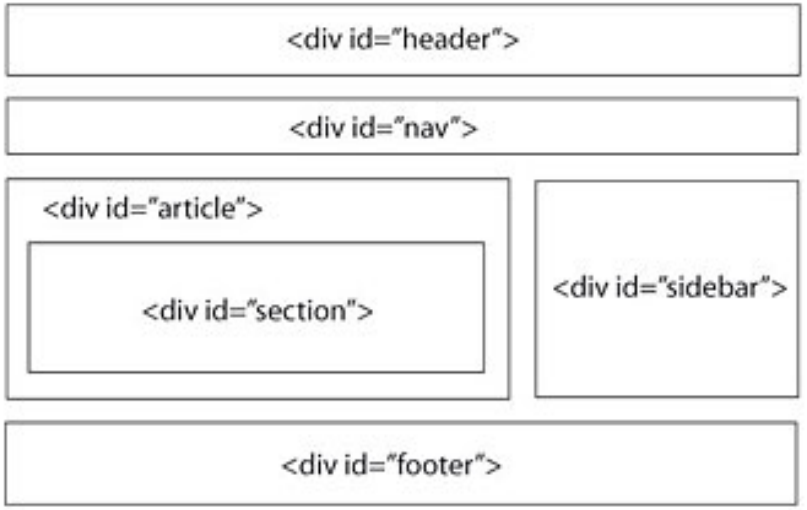
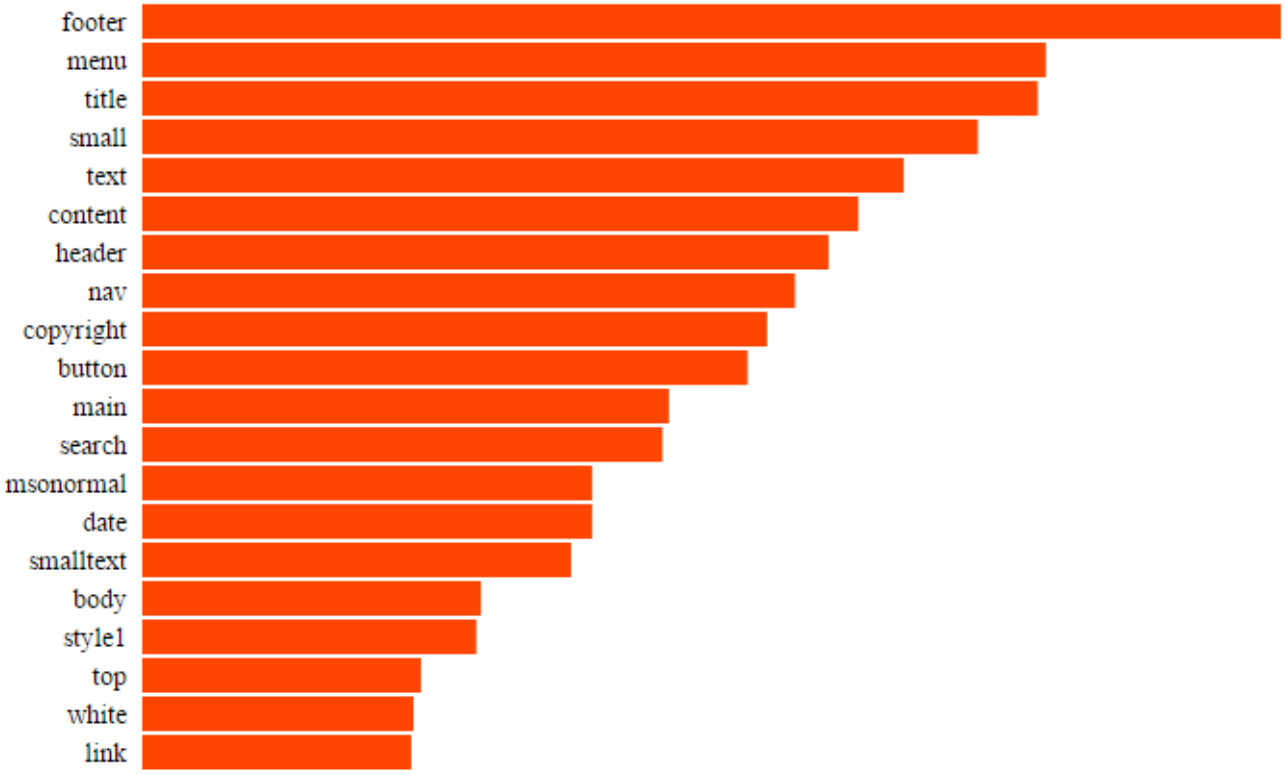


Fig 4. Non-semantic HTML, using generic divs

Из истории HTML5

- Имена новых элементов были взяты из 20 наиболее популярных имен параметров id и class



https://developers.google.com/webmasters/state-of-the-web/2005/classes

Из истории HTML5

Popular Class	HTML5 Element
footer	footer
menu	menu
title, header, top (?)	header
small, smalltext	small
text, content, main, body	article
nav	nav
copyright	*none yet*
button	*working around an IE6 limitation*
search	*none yet*
date	date
link	*?*

Graceful Degradation

(Постепенное сокращение, изящная деградация)
Подход к разработке, когда в первую очередь создается продукт, ожидающий от среды полный набор компонентов, после чего добавляется возможность работы при отсутствии всех или некоторых из них.

- Если начинать создание сайта под новейшие браузеры, а потом просто «урезать» некоторые краеугольные особенности для более старых — это принцип graceful degradation. (Placeholder)

Progressive Enhancement

(Прогрессивное улучшение) Противоположный по отношению к graceful degradation подход. Изначально продукт разрабатывается исходя из выполнения в ограниченной среде, а затем функционал постепенно наращивается по мере доступности компонентов.

- Если начинать верстку с самого старого, но всё ещё общеиспользуемого браузера, а после добавлять некоторые особенности для более новых версий, то это — progressive enhancement (Текст в поле)

Browser centric

Браузер важнее пользователей:

Например сообщение

«Скачайте самую последнюю версию браузера»

Это не Graceful Degradation

СЕМАНТИЧЕСКАЯ ВЕРСТКА

Семантическая верстка

- Семантика в верстке – это соответствие тегов к информации находящейся внутри них. Семантика кода также достигается путем уменьшения количества тегов. Таким образом, мы создаем чистый, читабельный, валидный HTML код. Такая страница будет быстрее грузиться и ранжироваться поисковыми системами.
- **HTML-разметка и названия CSS-классов должны отражать не оформление, а смысл.**

Семантическая вёрстка

Неправильно

- `<div style="color:red; border: 1px solid red">Плохая вёрстка сообщения об ошибке: атрибут style! </div>`

- `<div class="red red-border">Плохая вёрстка сообщения об ошибке: несемантический class! </div>`

Правильно

- `<div class="error">Сообщение об ошибке (error), правильная вёрстка! </div>`

Семантическая верстка

Semantic HTML

```
<h5>Benefits</h5>
<p>There are five main benefits of
 using semantic HTML:</p>
<ul>
    <li>Lighter code</li>
    <li>Maintainable code</li>
    <li>Accessibility</li>
    <li>Search visibility</li>
    <li>Future-proofing</li>
</ul>
```

Output:

Benefits

There are five main benefits of using semantic HTML:

- Lighter code
- Maintainable code
- Accessibility
- Search visibility
- Future-proofing

Fig 1. A heading, a paragraph, and a list

Non-semantic HTML

```
<p><b>Benefits</b></p>
There are five main benefits of usi
ng semantic HTML:
<p>
<img src="o.jpg">Lighter code<br>
<img src="o.jpg">Maintenance<br>
<img src="o.jpg">Accessibility<br>
<img src="o.jpg">SEO<br>
<img src="o.jpg">Future-proofing
</p>
```

Output:

Benefits

There are five main benefits of using semantic HTML:

- Lighter code
- Maintainable code
- Accessibility
- Search visibility
- Future-proofing

Fig 2. Bold text, paragraphs, and line-breaks

Семантическая верстка

- Заголовки должны выделяться тегами H1, H2, H3, H4, но никак не B и STRONG.
- При создании меню лучше всего использовать UL список, внутри которого будут лежать LI элементы меню. Этим мы показываем, что ссылки равносильные. Если имеются пункты второй вложенности, соответственно создаем внутри первичного LI элемента еще один UL список.
- Все служебные картинки (иконки, стрелки, ...) должны быть прописаны в CSS коде. В HTML, тег IMG должен использоваться только для больших картинок. (размер более 100 x 100 px)

Семантическая верстка

- Параграф блока текста создается с помощью P тега, но никак не DIV.
- Не использовать атрибуты STYLE * внутри HTML тега. Все стили выносить в отдельный CSS файл.
- То же самое по поводу JavaScript *.
- Соблюдать иерархию и логику документа. Более важные элементы страницы должны стоять в начале HTML кода, менее в конце. С помощью CSS стилей и DIV блоков, этого достичь не сложно, при любой схеме шаблона.

- * - кроме «первого экрана»

Семантическая верстка

- **Удаляем ненужные div теги**

```
<div class="contactform">
  <form>
     ....
  </form>
</div>
```

=

```
<form class="contactform">
     ....
</form>
```

```
div.contactform {
  margin: 20px 10px;
  border: solid 1px #ccc;
  background: #fff;
}
```

```
.contactform {
  margin: 20px 10px;
  border: solid 1px #ccc;
  background: #fff;
}
```

```
<div id="sidebar">
  <div class="box">
    <h4>Heading</h4>
    content
    ...
  </div>
  <div class="box">
    <h4>Heading</h4>
    content
    ...
  </div>
</div>
```

=

```
<div id="sidebar">
  <h4>Heading</h4>
  content
  ...
  <h4>Heading</h4>
  content
  ...
</div>
```

Семантическая верстка

- **Правильная разметка HTML кода**

```
<div>Heading here</div>
<div>Posted in: <a href="#">Web Design</a></div>
<div>Content here...</div>
```

Heading here
Posted in: Web Design
Content here...

```
<h2>Heading here</h2>
<p>Posted in: <a href="#">Web Design</a></p>
<p>Content here...</p>
```

Heading here

Posted in: Web Design

Content here...

Полезные ресурсы

http://htmlbook.ru/layout

https://developers.google.com/web/fundamentals/design-and-ui/responsive/patterns?hl=ru

http://www.w3schools.com/html/html_layout.asp

http://ru.learnlayout.com/

http://www.htmldog.com/guides/css/intermediate/layout/

https://html5book.ru/category/vyorstka/

http://codereview.stackexchange.com/questions/22832/html5-semantic-layout-div-article-section-and-explicit-identifiers