



Передовые  
инженерные  
школы



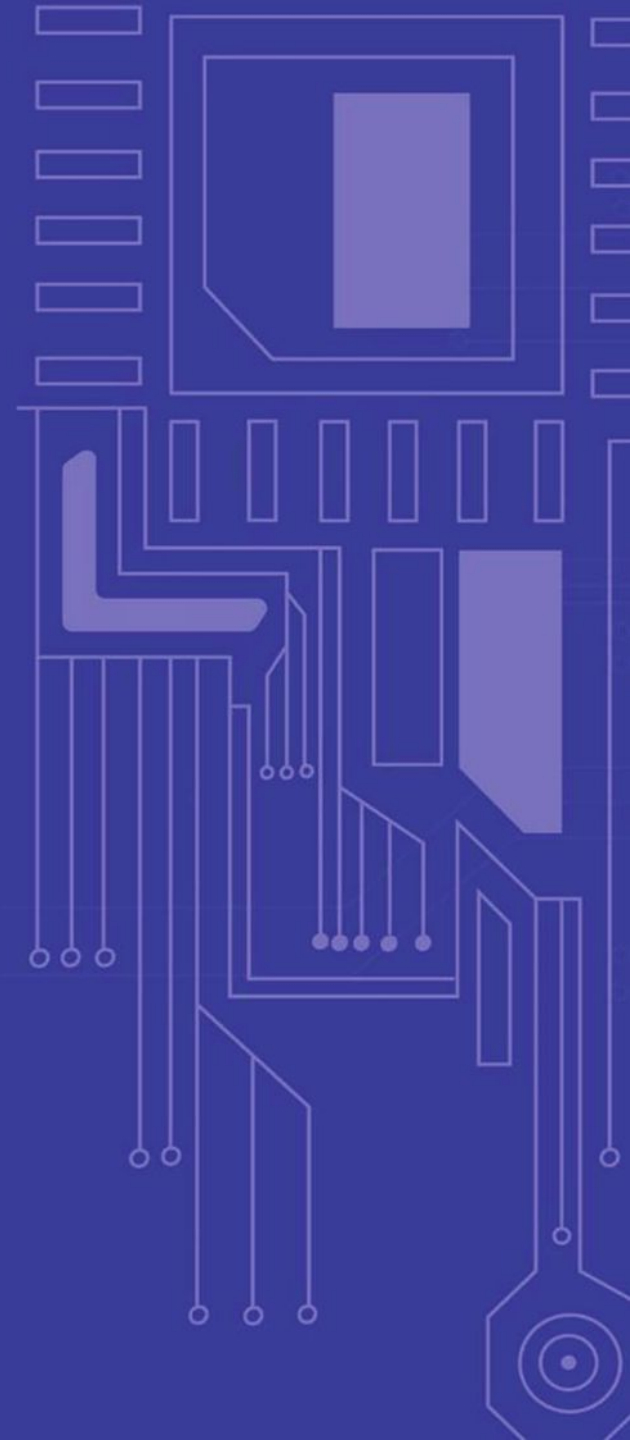
МИНОБРНАУКИ  
РОССИИ



УНИВЕРСИТЕТ  
ИННОПОЛИС

# Занятие 11

## Responsive Design

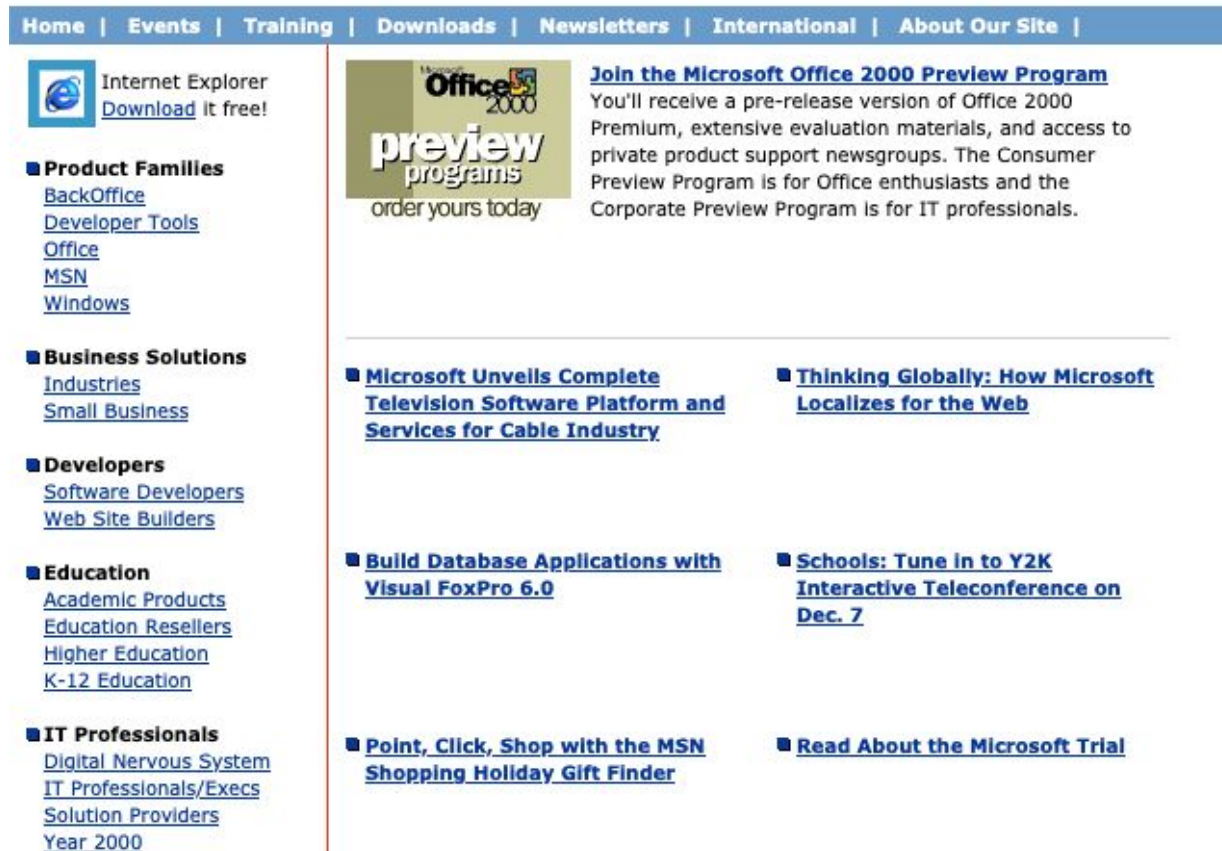


# План занятия

1. Введение
2. Дизайн и его примеры

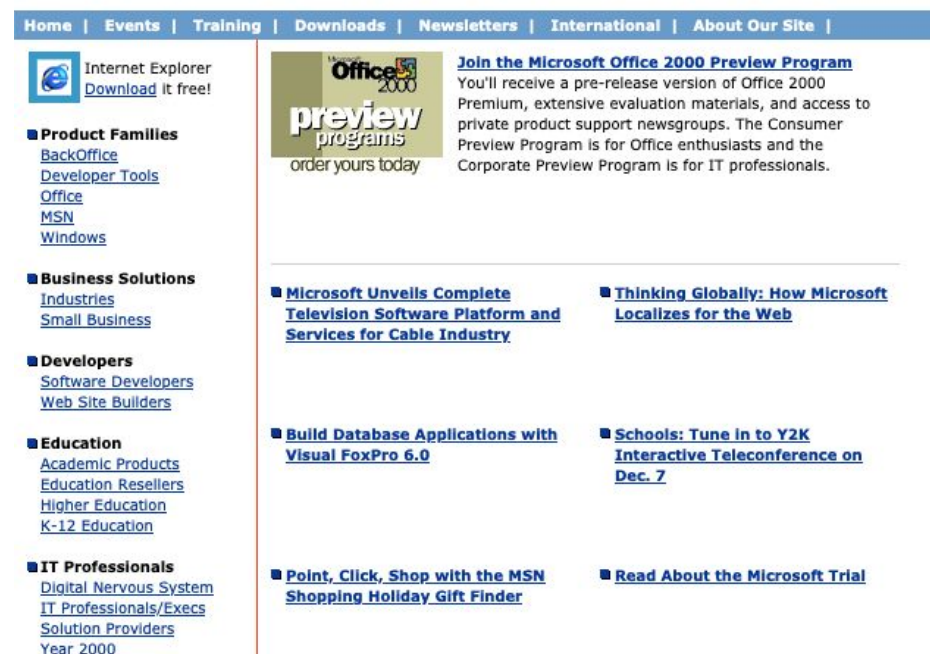
Responsive Design!

Адаптивный (Responsive) дизайн - не первый подход к разработке веб-сайтов. За годы до появления адаптивного дизайна веб-дизайнеры и разработчики испробовали множество различных методов.



# Дизайн с фиксированной шириной

В начале 1990-х, когда Интернет только набирал популярность, большинство мониторов имели размеры экрана 640 пикселей в ширину и 480 пикселей в высоту. Это были выпуклые электронно-лучевые трубки, не похожие на плоские жидкокристаллические дисплеи, которые мы имеем сейчас.

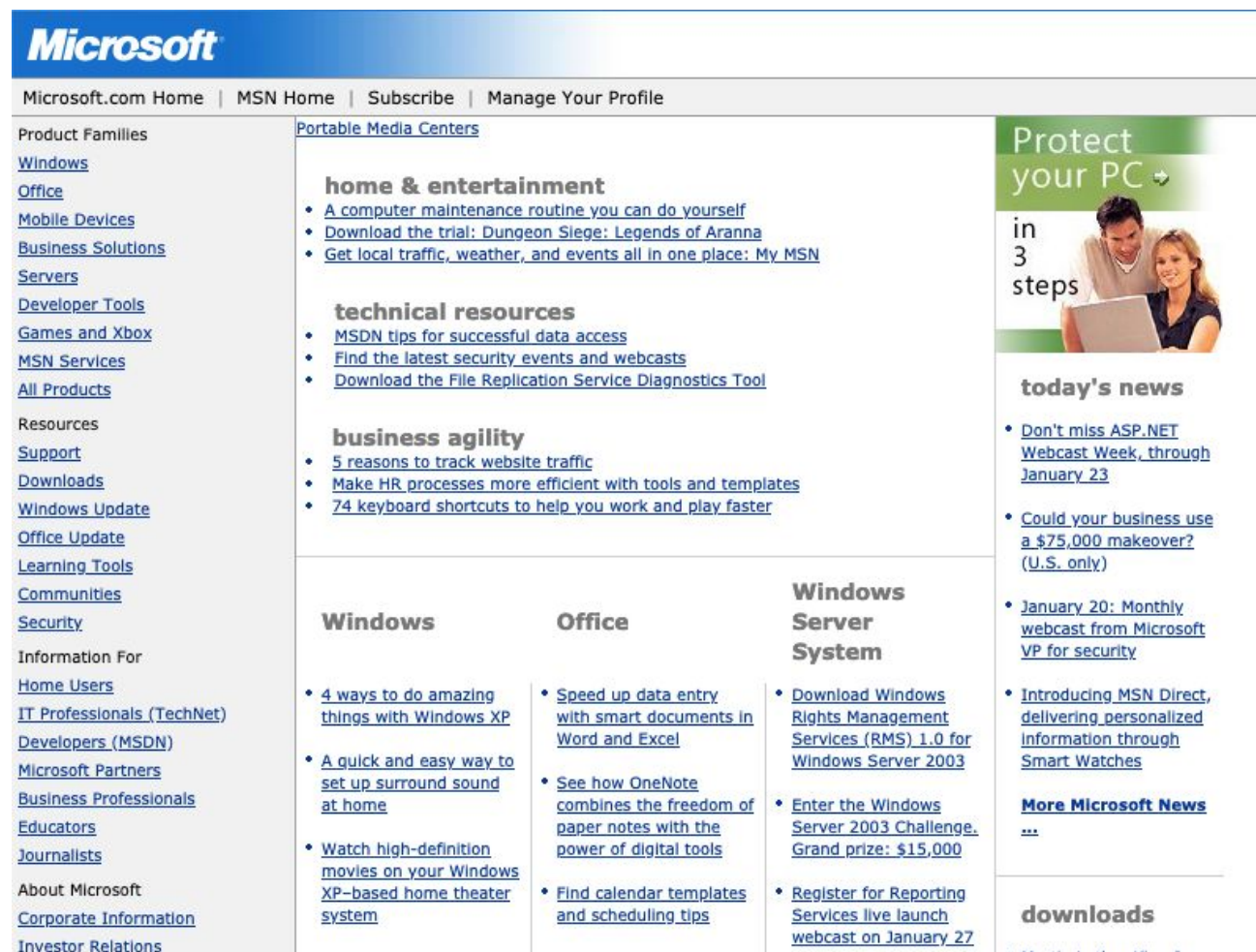


# Дизайн с фиксированной шириной



ПЕРЕДОВАЯ  
ИНЖЕНЕРНАЯ ШКОЛА  
УНИВЕРСИТЕТА ИННОПОЛИС

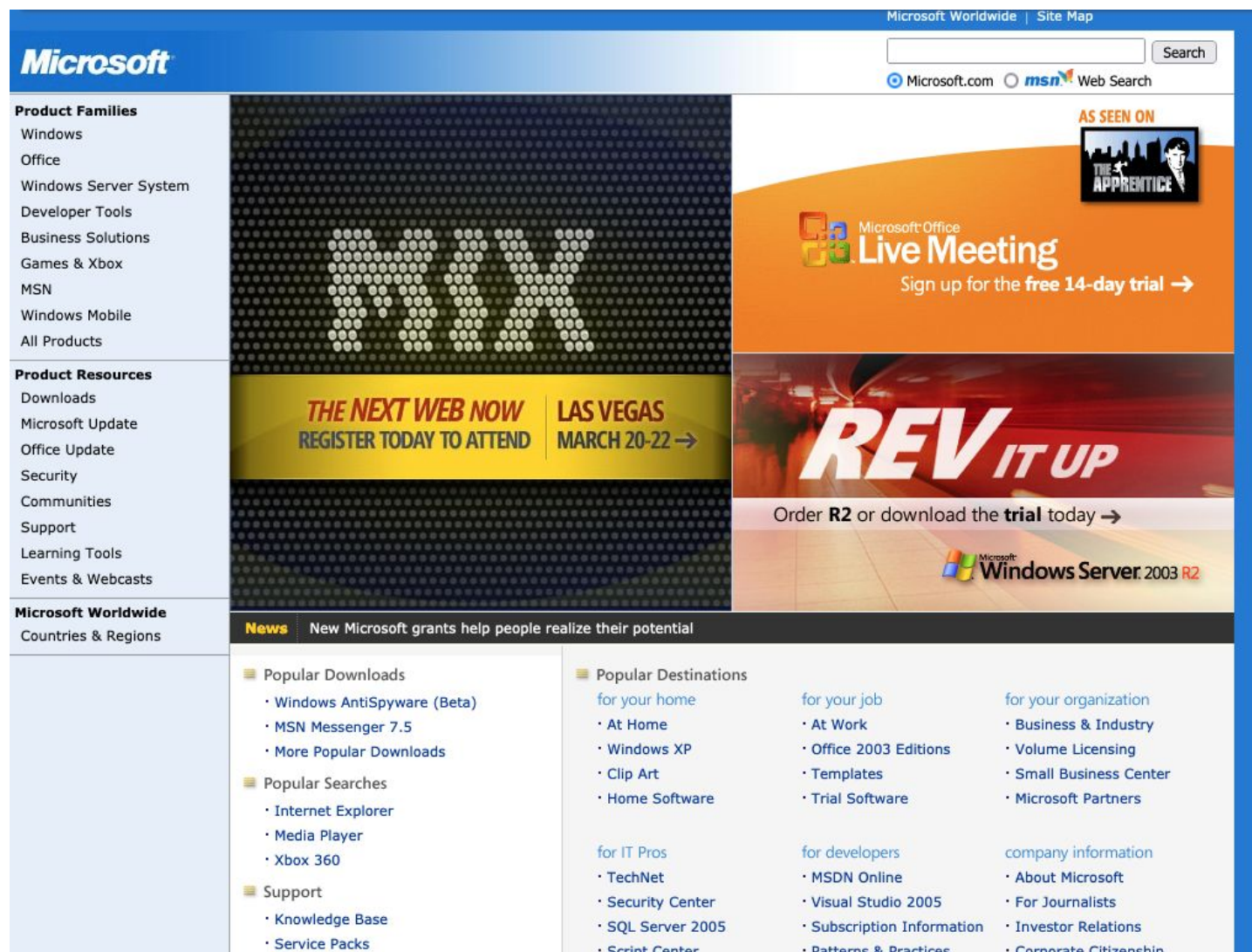
В дни становления раннего веб-дизайна было безопасно создавать веб-страницы шириной в 640 пикселей. Но в то время как другие технологии, такие как телефоны и камеры, миниатюризировались, экраны становились больше (и, в конечном счете, более плоскими). Вскоре большинство экранов имели размеры 800 на 600 пикселей. Соответственно изменился веб-дизайн. Дизайнеры и разработчики начали предполагать, что 800 пикселей являются безопасным значением по умолчанию.





# Дизайн с фиксированной шириной

Затем экраны снова стали больше. по умолчанию было установлено разрешение 1024 на 768. Это было похоже на гонку вооружений между веб-дизайнерами и производителями оборудования.



# Дизайн с фиксированной шириной

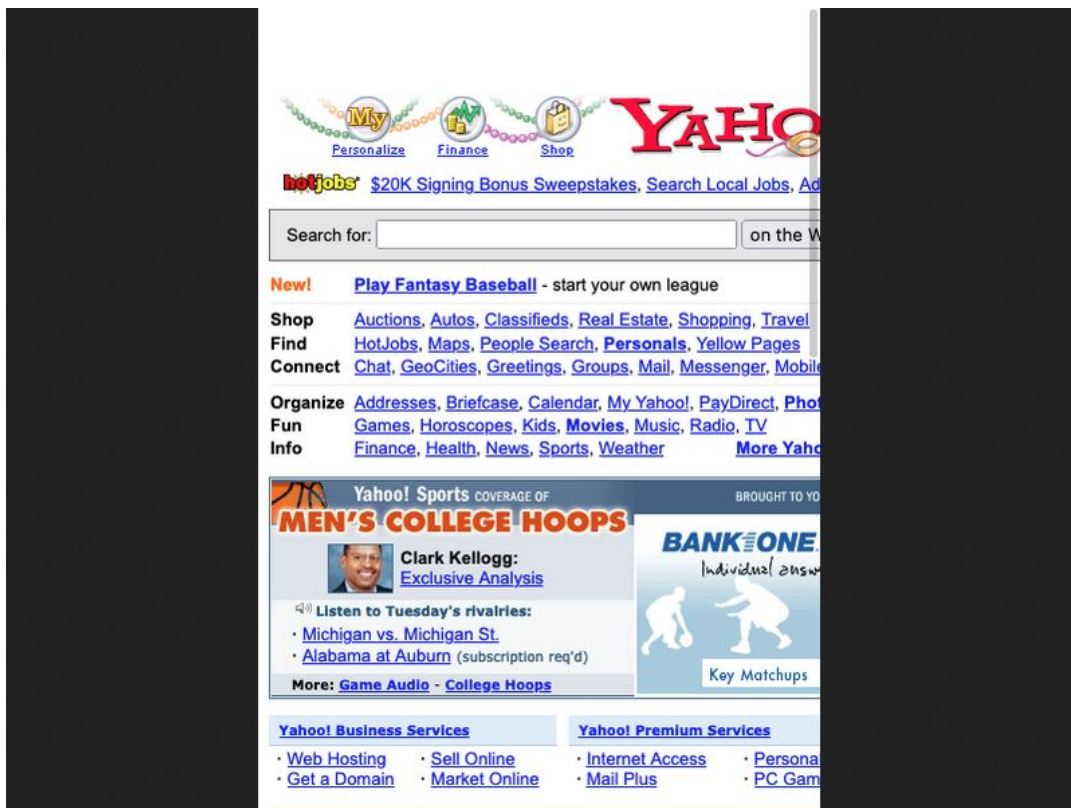
Независимо от того, было ли это 640, 800 или 1024 пикселя, выбор одной конкретной ширины для проектирования назывался дизайном с фиксированной шириной.

- Если вы укажете фиксированную ширину для своего макета, то ваш макет будет хорошо смотреться только при этой конкретной ширине.
- Если у посетителя вашего сайта экран шире, чем выбранная вами ширина, то на экране будет потрачено впустую пространство. Вы можете расположить содержимое своих страниц по центру, чтобы распределить это пространство более равномерно (вместо того, чтобы оставлять пустое место с одной стороны), но вы все равно не сможете в полной мере использовать доступное пространство.



# Дизайн с фиксированной шириной

Аналогично, если посетитель приходит с более узким экраном, чем выбранная вами ширина, то ваш контент не будет помещаться по горизонтали. Браузер генерирует панель — горизонтальную полосу прокрутки — и пользователю приходится перемещать всю страницу влево и вправо, чтобы увидеть все содержимое.





# Дизайн с фиксированной шириной

В коде это выглядит так:

## HTML

```
<h1>Fixed-width design</h1>
<article>
  <p>If you specify a fixed width for your layout, then your
  layout will only look good at that specific width. If a
  visitor to your site has a screen that is wider than the
  width you have specified, then there'll be wasted space on
  the screen. You can center the content of your pages to
  distribute that space more evenly (instead of having empty
  space on one side) but you still wouldn't be taking full
  advantage of the available space.</p>
  <p>Similarly, if the visitor arrives with a screen that is
  narrower than the width you've chosen, then your content
  won't fit horizontally. The browser generates a scrollbar—the
  horizontal equivalent of a scrollbar—and the user has to
  move the whole page left and right in order to see all of
  the content.</p>
</article>
```

## CSS

```
body {
  width: 640px;
  font-family: sans-serif;
  line-height: 1.5;
  padding: 0 16px;
}

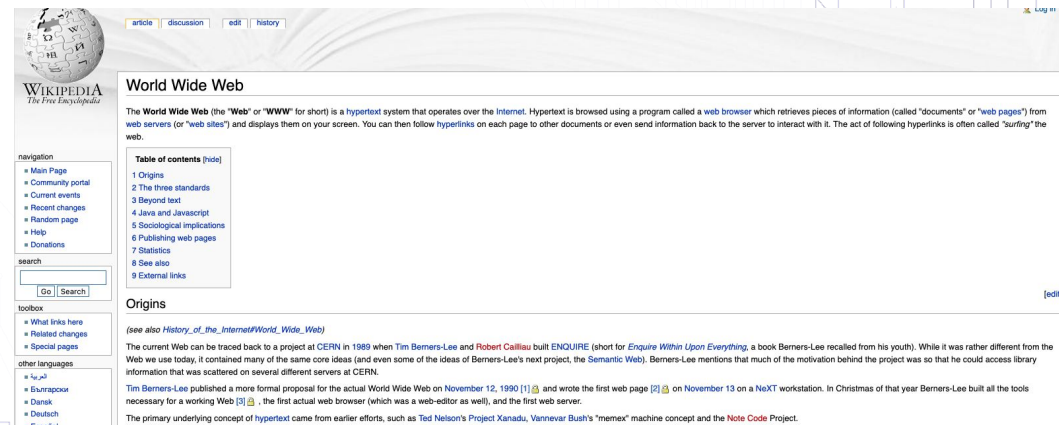
h1 {
  margin-bottom: 0;
}
```

# Жидкий дизайн (Liquid design)

Большинство дизайнеров использовали макеты фиксированной ширины, некоторые предпочли сделать свои макеты гибкими. Вместо использования фиксированной ширины для своих макетов вы могли бы создать гибкий макет, используя проценты для ширины столбцов. Эти дизайны работают в большем количестве ситуаций, чем макет фиксированной ширины, который выглядит правильно только при одном определенном размере.

**Это жидкими макетами.** Но, хотя жидкий макет будет хорошо смотреться в широком диапазоне ширины, в крайних случаях он начнет ухудшаться.

- На широком экране макет выглядит растянутым.
- На узком экране макет выглядит сжатым. Оба сценария не идеальны.



# Жидкий дизайн (Liquid design)

Устранить эти проблемы можно используя min-width и max-width для макета.

Но тогда при любом размере ниже минимальной ширины или выше максимальной возникают те же проблемы, что и при макете фиксированной ширины.

- На широком экране неиспользуемое пространство будет потрачено впустую.
- На узком экране пользователю пришлось бы перемещать всю страницу влево и вправо, чтобы увидеть все.

Слово "жидкий" - это всего лишь один из терминов, используемых для описания такого рода макетов. Подобные проекты также назывались **текущими макетами** или **гибкими макетами**. Терминология была такой же изменчивой, как и техника.

# Маленькие экраны

В коде это так:

```
<h1>Liquid layout</h1>
<article>
  <p>Instead of using fixed widths for your layouts you could make a
flexible layout using percentages for your column widths. This will
work in more situations than a fixed-width layout that only looks
right at one specific size. These were called liquid layouts.</p>
  <p>But while a liquid layout will look good across a wide range of
widths, it will begin to worsen at the extremes. On a wide screen the
layout will look like it's been stretched out too far. On a narrow
screen the layout will look like it's been squashed. Both scenarios
feel uncomfortable.</p>
  <p>You can mitigate these problems by using <code>min-width</code>
and <code>max-width</code> for your layout. But then at any sizes
below the minimum width or above the maximum width, you've got the same
issues you'd have with a fixed-width layout. On a wide screen there'd
be unused space going to waste. On a narrow screen, the user would
have to move the whole page left and right in order to see
everything.</p>
</article>
<aside>
  <p>The word "liquid" is just one of the terms used to describe this
kind of layout. These kinds of designs were also called fluid layouts
or flexible layouts. The terminology was as fluid as the
technique.</p>
  <p>This example is using the CSS <code>float</code> property to
create columns. That was a popular technique before CSS grid or
flexbox existed.</p>
</aside>
```

```
body {
  font-family: sans-serif;
  line-height: 1.5;
  padding: 0 16px;
}
```

```
article {
  width: 66%;
  float: left;
}
```

```
aside {
  width: 33%;
  float: right;
}
```

```
h1 {
  margin-bottom: 0;
}
```



# Маленькие экраны

Web продолжал становиться все больше и больше. То же самое происходило и с мониторами.

Но появились новые экраны, которые были меньше любого настольного устройства.

С появлением мобильных телефонов с полнофункциональными веб-браузерами дизайнеры столкнулись с дилеммой.

Как они могли гарантировать, что их дизайн будет хорошо смотреться на настольном компьютере и мобильном телефоне?

Им нужен был способ оформления своего контента для экранов шириной всего 240 пикселей и шириной в тысячи пикселей

# Отдельные сайты

Один из вариантов - создать отдельный поддомен для мобильных посетителей.

Но тогда вам придется поддерживать две отдельные базы кода и дизайны.

И для того, чтобы перенаправлять посетителей на мобильных устройствах, вам нужно было бы выполнить определение устройства пользователя и user-agent, который может быть ненадежным и его легко подделать.

Chrome будет отказываться от своей строки user-agent по соображениям конфиденциальности.

Кроме того, нет четкой границы между мобильными и немобильными.

На какой сайт вы отправляете планшетные устройства?

# Адаптивные макеты

Вместо того, чтобы иметь отдельные сайты на разных поддоменах, можно сделать один сайт с двумя или тремя макетами фиксированной ширины.

Медиа-запросы впервые появились в CSS, они открыли путь к повышению гибкости макетов.

Но многим разработчикам по-прежнему было удобнее всего создавать макеты фиксированной ширины.

Один из методов включал переключение между несколькими макетами фиксированной ширины с заданной шириной.

Некоторые люди называют это адаптивным дизайном.

# Адаптивные макеты

Адаптивный дизайн позволял дизайнерам создавать макеты, которые хорошо выглядели при нескольких разных размерах.

Но дизайн никогда не выглядел совершенно правильно при просмотре между этими размерами.

Проблема избыточного пространства сохранялась, хотя она была не такой серьезной, как при макете фиксированной ширины.

Используя мультимедийные запросы CSS, можно предоставить пользователям макет, максимально приближенный к ширине их браузера.

Но, учитывая разнообразие размеров устройств, скорее всего, макет будет выглядеть не совсем идеально для большинства людей.



# Адаптивные макеты

В коде это сделано так:

```
<h1>Adaptive design</h1>
<article>
  <p>When media queries arrived in CSS it opened the door to
  making layouts more flexible. But developers were still most
  comfortable making fixed-width layouts. One technique
  involved switching between a handful of fixed-width layouts
  at specified widths.</p>
  <p>This allowed designers to provide layouts that looked
  good at a few different sizes but the design never looked
  quite right when viewed <em>between</em> those sizes. The
  problem of excess space persisted although it wasn't as bad
  as in a fixed-width layout.</p>
</article>
<aside>
  <p>Ultimately this technique wasn't very popular. The term
  "adaptive" was also used to refer to other approaches so it
  can be a confusing descriptor for what was quite a niche
  technique.</p>
  <p>This example is using the CSS <code>float</code>
  property to create columns. That was a popular technique
  before CSS grid or flexbox existed.</p>
</aside>
```

# Адаптивные макеты

В коде это сделано так:

```
body {  
  font-family: sans-serif;  
  line-height: 1.5;  
  padding: 0 16px;  
}  
  
h1 {  
  margin-bottom: 0;  
}  
  
@media all and (min-width: 800px) {  
  article {  
    width: 540px;  
    float: left;  
  }  
  
  aside {  
    width: 250px;  
    float: left;  
    margin-left: 10px;  
  }  
}  
  
@media all and (min-width: 1200px) {  
  article {  
    width: 800px;  
    float: left;  
  }  
  
  aside {  
    width: 350px;  
    float: left;  
    margin-left: 50px;  
  }  
}  
  
@media all and (min-width: 1600px) {  
  h1 {  
    width: 400px;  
    float: left;  
  }  
  
  article {  
    width: 800px;  
    float: left;  
  }  
  
  aside {  
    width: 350px;  
    float: left;  
    margin-left: 50px;  
  }  
}
```

# Адаптивные web-design

Если адаптивные макеты – это сочетание медиа-запросов и макетов фиксированной ширины, то адаптивный веб-дизайн – это сочетание медиа-запросов и гибких макетов.

```
<h1>Responsive design</h1>
<article>
  <p>If adaptive layouts are a mashup of media queries and fixed-width layouts,
responsive web design is a mashup of media queries and liquid layouts. The term
was coined by Ethan Marcotte in <a
href="https://alistapart.com/article/responsive-web-design/">an article in A
List Apart</a> in 2010.</p>
  <p>Ethan defined three criteria for responsive design:</p>
  <ol>
    <li>Fluid grids</li>
    <li>Fluid media</li>
    <li>Media queries</li>
  </ol>
  <p>If a site was responsive, its layout and images would look good on any
device.</p>
</article>
<aside>
  <p>This example is using the CSS <code>float</code> property to create
columns. That was a popular technique before CSS grid or flexbox existed.</p>
</aside>
```

# Адаптивные web-design

Если адаптивные макеты – это сочетание медиа-запросов и макетов фиксированной ширины, то адаптивный веб-дизайн – это сочетание медиа-запросов и гибких макетов.

```
body {  
  font-family: sans-serif;  
  line-height: 1.5;  
  padding: 0 16px;  
}
```

```
h1 {  
  margin-bottom: 0;  
}
```

```
@media all and (min-width: 800px) {  
  article {  
    width: 66%;  
    float: left;  
  }
```

```
  aside {  
    width: 33%;  
    float: right;  
  }  
}
```

```
@media all and (min-width: 1600px) {  
  h1 {  
    width: 22.5%;  
    float: left;  
  }
```

```
  article {  
    width: 50%;  
    float: left;  
  }
```

```
  aside {  
    width: 22.5%;  
    float: right;  
  }  
}
```



# Адаптивные web-design

Три критерия для адаптивного дизайна:

1. Жидкостные решетки (Fluid grids)
2. Текучие среды (Fluid media)
3. Медиа-запросы (Media queries)

Макет и изображения адаптивного сайта будут хорошо смотреться на любом устройстве.

# Meta – элемент viewport

Браузерам на мобильных телефонах приходилось иметь дело с веб-сайтами, которые были разработаны с макетами фиксированной ширины для более широких экранов.

По умолчанию мобильные браузеры предполагали, что люди проектировали ширину в 980 пикселей (и они не ошибались).

Таким образом, даже если бы вы использовали жидкий макет, браузер применил бы ширину в 980 пикселей, а затем уменьшил бы отображаемую веб-страницу до фактической ширины экрана.

Используя адаптивный дизайн, нужно указать браузеру не выполнять такое масштабирование. Для этого задаем meta в head:

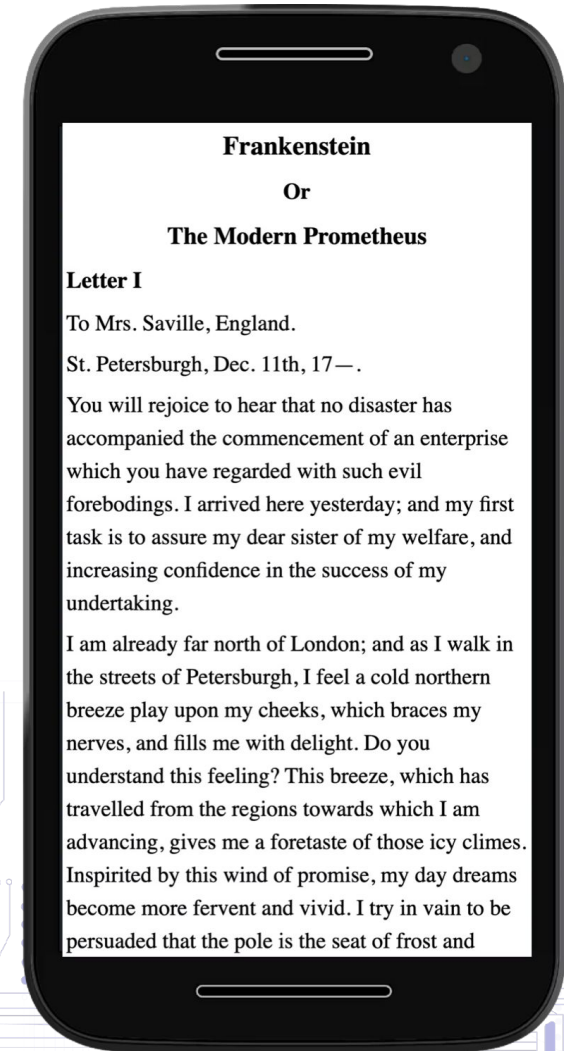
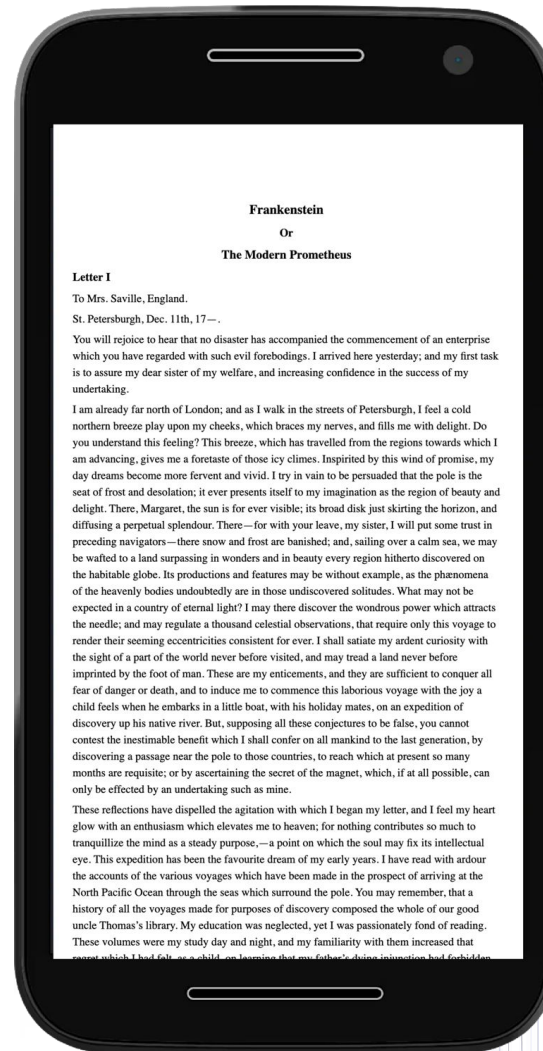
```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

# Meta – элемент viewport

Существует два значения:

- `width=device-width` - указывает браузеру предположить, что ширина веб-сайта совпадает с шириной устройства (вместо предположения, что ширина веб-сайта равна 980 пикселям)
- `initial-scale=1` - Указывает браузеру, какое масштабирование требуется выполнить

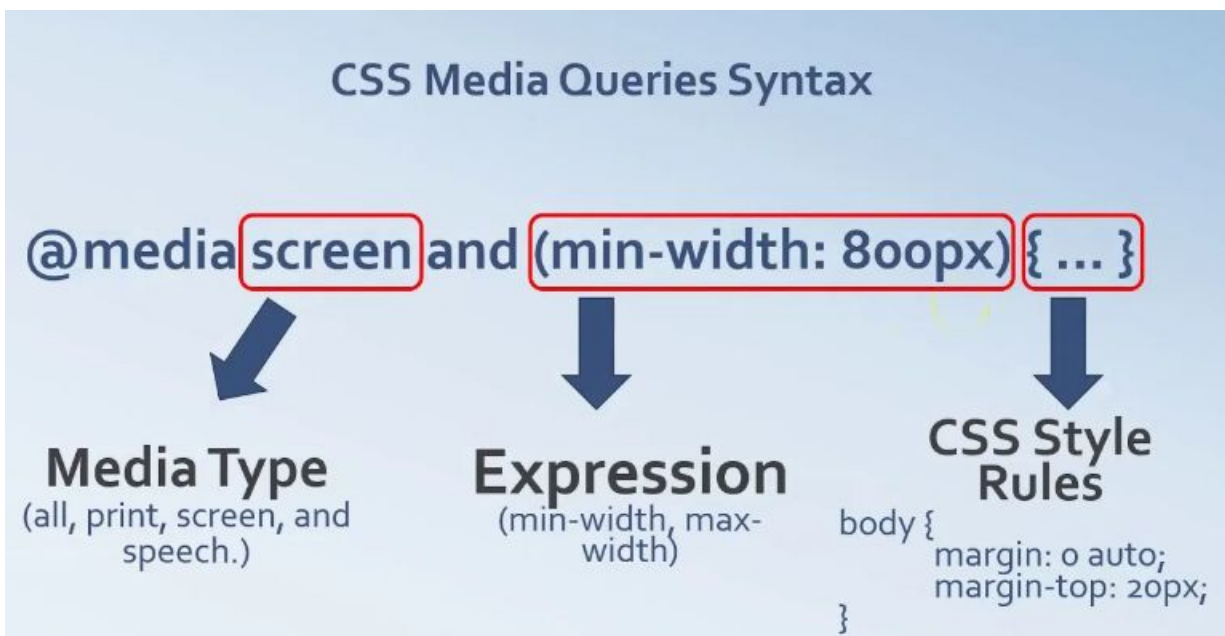
При адаптивном дизайне мы не хотим чтобы браузер выполнял какое-либо масштабирование.



# Медиа-запросы

Помогают адаптировать дизайны к различным размерам экрана с помощью медиа-запросов CSS

Медиа-запросы инициируются с помощью **@media** ключевого слова (CSS at-правила) и могут использоваться для различных вариантов использования.





Веб-сайты часто отображаются на экранах, но CSS также можно использовать для оформления веб-сайтов и для других выходных данных.

При распечатке.

В этом примере цвет фона установлен на серый. Но если страница печатается, цвет фона должен быть прозрачным. Это экономит чернила пользовательского принтера.

```
body {  
    color: black;  
    background-color: grey;  
}  
  
@media print {  
    body {  
        background-color: transparent;  
    }  
}
```

# Медиа-запросы

Можно использовать @media правило в своей таблице стилей подобным образом, или создать отдельную таблицу стилей и использовать media с атрибутом link

```
<link rel="stylesheet" href="global.css">  
<link rel="stylesheet" href="print.css" media="print">
```

Если не указать какой-либо тип мультимедиа для CSS, он автоматически получит значение типа мультимедиа all

Эти два блока CSS эквивалентны:

```
body {  
  color: black;  
  background-color: white;  
}
```

```
@media all {  
  body {  
    color: black;  
    background-color: white;  
  }  
}
```

```
<link rel="stylesheet" href="global.css">
```

```
<link rel="stylesheet" href="global.css" media="all">
```

Для применения на определенном устройстве можно добавлять условия к типам носителей.

CSS применяется, только если тип носителя совпадает и условие также истинно.

Эти условия называются *функциями носителя* (media features)

Это синтаксис для медиа-запросов:

`@media type and (feature)`

Если стили находятся в отдельной таблице стилей используем link:

```
<link rel="stylesheet" href="specific.css" media="type and (feature)">
```

Допустим, хотим применить разные стили в зависимости от того, находится ли окно браузера в альбомном режиме (ширина области просмотра больше ее высоты) или в портретном режиме (высота области просмотра больше ее ширины).

Есть медиа-функция под названием **orientation**, которую вы можете использовать для тестирования этого:

```
@media all and (orientation: landscape) {  
    // Styles for landscape mode.  
}  
@media all and (orientation: portrait) {  
    // Styles for portrait mode.  
}
```

```
<link rel="stylesheet" href="landscape.css" media="all and (orientation: landscape)">  
<link rel="stylesheet" href="portrait.css" media="all and (orientation: portrait)">
```



# Медиа-запросы

## <h1>Combining media queries</h1>

<p>You can use lots of different CSS units in your media queries. If your content is mostly image-based, pixels might make the most sense. If your content is mostly text-based, it probably makes more sense to use a unit that's based on text size, like ``rem`` or ``em``</p>

<p>You can also combine media queries to apply more than one condition. Use the word ``and`` to combine your media queries.</p>

<p>In this example, the background color will change but only when the viewport width is between `<code>50em</code>` and `<code>60em</code>`.</p>

```
body {
  margin: var(--metric-box-spacing);
  background-color: color: var(--color-light);
  color: var(--color-dark);
}
:is(h1,h2,h3,p,li) {
  max-inline-size: initial;
  margin-block-start: var(--metric-box-spacing);
}
code {
  color: var(--color-dark);
}
@media (min-width: 50em) and (max-width: 60em) {
  body {
    background-color: var(--color-mid-dark);
    color: var(--color-light);
  }
}
```

## Настройка стилей в зависимости от размеров окна просмотра

Для адаптивного дизайна одной из наиболее полезных мультимедийных функций являются размеры области просмотра браузера.

Чтобы применить стили, когда окно браузера шире определенной ширины используйте **min-width**:

```
@media (min-width: 400px) {  
    // Styles for viewports wider than 400 pixels.  
}
```

Используйте **max-width** функцию мультимедиа, чтобы применить стили ниже определенной ширины:

```
@media (max-width: 400px) {  
    // Styles for viewports narrower than 400 pixels.  
}
```

В медиа-запросах можно использовать любые единицы длины CSS.

- Если контент в основном основан на изображениях, пиксели могут иметь наибольший смысл.
- Если ваш контент в основном основан на тексте, вероятно, имеет смысл использовать относительную единицу, основанную на размере текста

```
@media (min-width: 25em) {  
  // Styles for viewports wider than 25em.  
}
```

# Медиа-запросы

Можно комбинировать медиа-запросы, чтобы применить более одного условия. Используя слово **and**

```
@media (min-width: 50em) and (max-width: 60em) {  
    // Styles for viewports wider than 50em and narrower than  
    60em.  
}
```

```
<header>
<h1>Sketch of
The Analytical Engine
Invented by Charles Babbage </h1>
<h2>With notes upon the Memoir by the Translator Ada Augusta, Countess of Lovelace</h2>
</header>
<article>
<h3>Note C</h3>
<p> Those who may desire to study the principles of the Jacquard-loom in the most effectual manner, viz. that of practical observation, have only to step into the Adelaide Gallery or the Polytechnic Institution. In each of these valuable repositories of scientific illustration, a weaver is constantly working at a Jacquard-loom, and is ready to give any information that may be desired as to the construction and modes of acting of his apparatus. The volume on the manufacture of silk, in Lardner's Cyclopædia, contains a chapter on the Jacquard-loom, which may also be consulted with advantage.</p>
<p>The mode of application of the cards, as hitherto used in the art of weaving, was not found, however, to be sufficiently powerful for all the simplifications which it was desirable to attain in such varied and complicated processes as those required in order to fulfil the purposes of an Analytical Engine. A method was devised of what was technically designated backing the cards in certain groups according to certain laws. The object of this extension is to secure the possibility of bringing any particular card or set of cards into use any number of times successively in the solution of one problem. Whether this power shall be taken advantage of or not, in each particular instance, will depend on the nature of the operations which the problem under consideration may require. The process is alluded to by M. Menabrea, and it is a very important simplification. It has been proposed to use it for the reciprocal benefit of that art, which, while it has itself no apparent connexion with the domains of abstract science, has yet proved so valuable to the latter, in suggesting the principles which, in their new and singular field of application, seem likely to place algebraical combinations not less completely within the province of mechanism, than are all those varied intricacies of which intersecting threads are susceptible. By the introduction of the system of backing into the Jacquard-loom itself, patterns which should possess symmetry, and follow regular laws of any extent, might be woven by means of comparatively few cards.</p>
<p>Those who understand the mechanism of this loom will perceive that the above improvement is easily effected in practice, by causing the prism over which the train of pattern-cards is suspended to revolve backwards instead of forwards, at pleasure, under the requisite circumstances; until, by so doing, any particular card, or set of cards, that has done duty once, and passed on in the ordinary regular succession, is brought back to the position it occupied just before it was used the preceding time. The prism then resumes its forward rotation, and thus brings the card or set of cards in question into play a second time. This process may obviously be repeated any number of times. </p>
</article>
```



```
body {  
    margin: var(--metric-box-spacing);  
}  
  
:is(h1,h2,h3,p,li,article) {  
    max-inline-size: initial;  
    margin-block-start: var(--metric-box-spacing);  
}  
  
@media (min-width: 50em) {  
    article {  
        column-count: 2;  
    }  
}
```

# Контрольные точки для @media

Контрольная точка – точка, в которой условие функции мультимедиа становится истинным

Лучше всего выбирать точки на основе контента, а не популярных размеров устройств, поскольку они могут меняться с каждым циклом выпуска технологии.

Создавая точку нужно для того чтобы сделать интерфейс более разборчивым.

С помощью column-count свойства текст с этого момента разделяется на два столбца.

```
@media (min-width: 50em) {  
  article {  
    column-count: 2;  
  }  
}
```

```
body {  
  margin: var(--metric-box-spacing);  
}  
:is(h1,h2,h3,p,li,article) {  
  max-inline-size: initial;  
  margin-block-start: var(--metric-box-spacing);  
}  
@media (min-width: 50em) {  
  article {  
    column-count: 2;  
  }  
}
```

Мультимедийные запросы можно использовать на основе высоты окна просмотра, а не только ширины.

Это может быть полезно для оптимизации содержимого интерфейса "над сгибом".

В предыдущем примере, если читатели используют широкое, но короткое окно браузера, они должны прокрутить вниз один столбец, а затем прокрутить обратно вверх, чтобы добраться до начала второго столбца.

Было бы удобно применять столбцы только тогда, когда окно просмотра достаточно широкое и высокое.

# Комбинации @media

Мультимедийные запросы можно комбинировать таким образом, чтобы стили применялись только при выполнении всех условий.

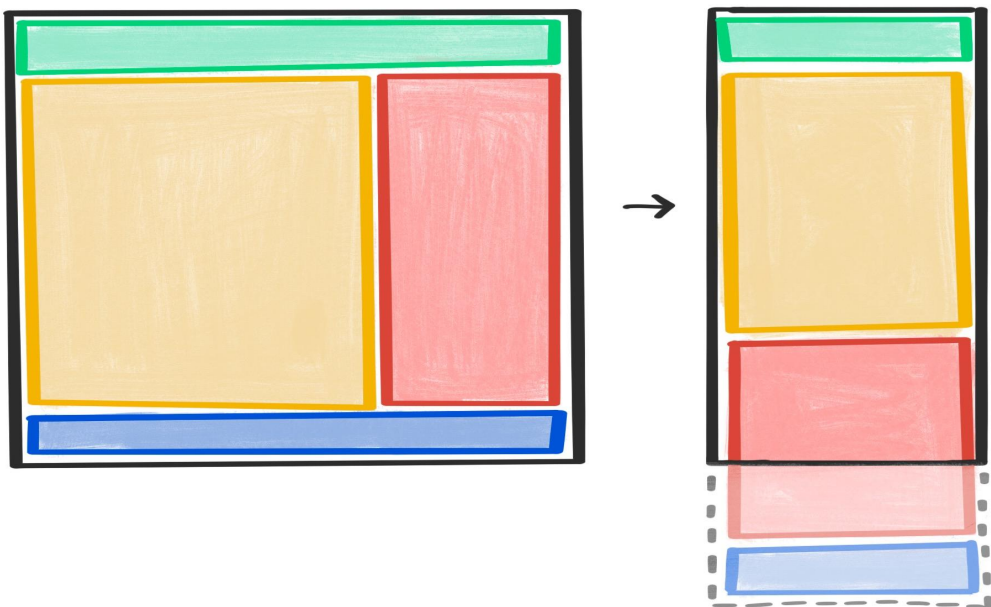
Окно просмотра должно быть как минимум 50em широким и 60em высоким, чтобы можно было применить стили столбцов.

Эти точки были выбраны на основе объема содержимого.

```
@media (min-width: 50em) and (min-height: 60em) {  
  article {  
    column-count: 2;  
  }  
}
```

Прежде чем применять какой-либо макет, нужно убедиться, что поток контента имеет смысл.

Этот порядок в одну колонку по умолчанию соответствует тому, что выводится экраны малого размера.



```
<body>  
  <header>...</header>  
  <main>  
    <article>...</article>  
    <aside>...</aside>  
  </main>  
  <footer>...</footer>  
</body>
```



```
<body>
  <header>
    My Website
  </header>
  <main>
    <article>
      <h1>My web page</h1>
      <p>This is my web page. There are many like it but this one is mine.</p>
      <p>I'm using the HyperText Markup Language (<dfn>HTML</dfn>) to give structure to this page.</p>
      <p>For example, this text is inside a paragraph which is inside an article which is inside the main
content.</p>
      <p>For the tangential information that follows, I'm using the <code>aside</code> element.</p>
    </article>
    <aside>
      <h2>History</h2>
      <p>Did you know that <a href="http://info.cern.ch/hypertext/WWW/TheProject.html">the first web page
ever made</a> is still online at its original URL?</p>
      <p>The page was created by Tim Berners-Lee in the early 90s and it still works in a modern web
browser.</p>
    </aside>
  </main>
  <footer>
    <small>I made this!</small>
  </footer>
</body>
```

```
body {  
  margin: 0;  
  padding: 0;  
}  
  
header,  
footer {  
  text-align: center;  
}  
  
header {  
  padding: var(--metric-box-spacing);  
  background-color: hsl(140deg 70% 95%);  
  font-size: 1.4em;  
}  
  
main {  
  min-height: calc(100vh - 7em);  
}  
  
article {  
  background-color: hsl(60deg 70% 95%);  
}
```

```
aside {  
  background-color: hsl(0deg 70% 95%);  
}  
  
article,  
aside {  
  padding: var(--metric-box-spacing);  
  margin: var(--metric-box-spacing);  
}  
  
footer {  
  padding: calc(var(--metric-box-spacing) / 2);  
  background-color: hsl(200deg 70% 95%);  
}  
  
p {  
  margin-block-start: var(--metric-box-spacing);  
}
```

После упорядочивания всех отдельных компонентов уровня страницы получаем высокоуровневый вид страницы



И только после этого...

Используя медиа-запросы, можно указать правила в CSS, описывающие, как этот вид должен адаптироваться к различным размерам экрана.

Grid - отличный инструмент для применения в макете страниц.

В приведенном выше примере, макет из двух столбцов, примениться как только будет доступна достаточная ширина экрана.

На гридах:

```
@media (min-width: 45em) {  
  main {  
    display: grid;  
    grid-template-columns: 2fr 1fr;  
  }  
}
```

```
<body>
  <header>
    My Website
  </header>
  <main>
    <article>
      <h1>My web page</h1>
      <p>This is my web page. There are many like it but this one is mine.</p>
      <p>I'm using the HyperText Markup Language (<dfn>HTML</dfn>) to give structure to this page.</p>
      <p>For example, this text is inside a paragraph which is inside an article which is inside the main
content.</p>
      <p>For the tangential information that follows, I'm using the <code>aside</code> element.</p>
    </article>
    <aside>
      <h2>History</h2>
      <p>Did you know that <a href="http://info.cern.ch/hypertext/WWW/TheProject.html">the first web page
ever made</a> is still online at its original URL?</p>
      <p>The page was created by Tim Berners-Lee in the early 90s and it still works in a modern web
browser.</p>
    </aside>
  </main>
  <footer>
    <small>I made this!</small>
  </footer>
</body>
```



# Grid



```
body {  
  margin: 0;  
  padding: 0;  
}  
  
p {  
  margin-block-start: var(--metric-box-spacing);  
}  
  
header,  
footer {  
  text-align: center;  
}  
  
header {  
  padding: var(--metric-box-spacing);  
  background-color: hsl(140deg 70% 95%);  
  font-size: 1.4em;  
}  
  
main {  
  min-block-size: calc(100vh - 6em);  
}
```

```
article {  
  background-color: hsl(60deg 70% 95%);  
}  
  
aside {  
  background-color: hsl(0deg 70% 95%);  
}  
  
article,  
aside {  
  padding: var(--metric-box-spacing);  
  margin: var(--metric-box-spacing);  
}  
  
footer {  
  padding: calc(var(--metric-box-spacing) / 2);  
  background-color: hsl(200deg 70% 95%);  
}  
  
@media (min-width: 45em) {  
  main {  
    display: grid;  
    grid-template-columns: 2fr 1fr;  
  }  
}
```

Для этого конкретного макета вы также могли бы использовать.

```
@media (min-width: 45em) {  
  main {  
    display: flex;  
    flex-direction: row;  
  }  
  
  main article {  
    flex: 2;  
  }  
  
  main aside {  
    flex: 1;  
  }  
}
```

Однако для версии flexbox требуется больше CSS. У каждого столбца есть отдельное правило, описывающее, сколько места он должен занимать. В примере с таблицей та же информация инкапсулирована в одно правило для содержащего элемента.

# Flex

```
body {  
  margin: 0;  
  padding: 0;  
}  
  
p {  
  margin-block-start: var(--metric-box-spacing);  
}  
  
header,  
footer {  
  text-align: center;  
}  
  
header {  
  padding: var(--metric-box-spacing);  
  background-color: hsl(140deg 70% 95%);  
  font-size: 1.4em;  
}  
  
main {  
  min-block-size: calc(100vh - 6em);  
}
```

```
article {  
  background-color: hsl(60deg 70% 95%);  
}  
  
aside {  
  background-color: hsl(0deg 70% 95%);  
}  
  
article,  
aside {  
  padding: var(--metric-box-spacing);  
  margin: var(--metric-box-spacing);  
}  
  
footer {  
  padding: calc(var(--metric-box-spacing) / 2);  
  background-color: hsl(200deg 70% 95%);  
}  
  
@media (min-width: 45em) {  
  main {  
    display: flex;  
    flex-direction: row;  
  }  
  main article {  
    flex: 2;  
  }  
  main aside {  
    flex: 1;  
  }  
}
```

# Когда использовать @media

Возможно не всегда нужно использовать медиа-запрос.

Если макет необходимо часто обновлять, мультимедийные запросы могут выйти из-под контроля с большим количеством точек.

Старайтесь использовать автоматический подход:

```
.cards {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, minmax(15em, 1fr));  
  grid-gap: 1em;  
}
```

# Когда использовать @media

```
<body>
  <header>
    My Website
  </header>
  <main>
    <h1>Cards</h1>
    <div class="cards">
      <article class="card">
        <h2>A card</h2>
        <p>This is a card. There are many like it but this one is mine.</p>
        <p>A card might have all sorts of content inside it. This one happens to have two paragraphs of text.</p>
      </article>
      <article class="card">
        <h2>A card</h2>
        <p>This is a card. There are many like it but this one is mine.</p>
        <p>A card might have all sorts of content inside it. This one happens to have two paragraphs of text.</p>
        <p>Actually, that's not true. This particular card has three paragraphs.</p>
      </article>
      <article class="card">
        <h2>A card</h2>
        <p>This is a card. There are many like it but this one is mine.</p>
        <p>A card might have all sorts of content inside it. This one happens to have two paragraphs of text.</p>
      </article>
      <article class="card">
        <h2>A card</h2>
        <p>This is a card. There are many like it but this one is mine.</p>
        <p>A card might have all sorts of content inside it. This one happens to have two paragraphs of text.</p>
      </article>
      <article class="card">
        <h2>A card</h2>
        <p>This is a card. There are many like it but this one is mine.</p>
        <p>A card might have all sorts of content inside it. This one happens to have two, no three, paragraphs of text.</p>
        <p>Notice how all the cards in a row automatically have the same height?</p>
      </article>
      <article class="card">
        <h2>A card</h2>
        <p>This is a card. There are many like it but this one is mine.</p>
        <p>A card might have all sorts of content inside it. This one happens to have two paragraphs of text.</p>
      </article>
    </div>
  </main>
  <footer>
    <small>I made this!</small>
  </footer>
</body>
```

# Когда использовать @media



```
body {  
  margin: 0;  
  padding: 0;  
}  
  
p {  
  margin-block-start: var(--metric-box-spacing);  
}  
  
h1 {  
  margin-block-end: var(--metric-box-spacing);  
}  
  
header,  
footer {  
  background-color: var(--color-off-white);  
  text-align: center;  
}  
  
header {  
  padding: var(--metric-box-spacing);  
  background-color: hsl(140deg 70% 95%);  
  font-size: 1.4em;  
}  
  
main {  
  padding: var(--metric-box-spacing);  
  min-block-size: calc(100vh - 8em);  
}  
  
footer {  
  padding: calc(var(--metric-box-spacing) / 2);  
  background-color: hsl(200deg 70% 95%);  
}  
  
.cards {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, minmax(15em, 1fr));  
  grid-gap: var(--metric-box-spacing);  
}  
  
.card {  
  padding: var(--metric-box-spacing);  
  background-color: var(--color-off-white);  
  border-radius: var(--metric-radius);  
  box-shadow: var(--generic-shadow);  
}
```



# Когда использовать @media

Аналогичную компоновку можно создать с помощью flexbox.

В этом случае, если карточек недостаточно для заполнения последней строки, оставшиеся карточки будут растягиваться, заполняя доступное пространство, а не выстраиваться в столбцы.

Если вы хотите выровнять строки и столбцы, используйте grid.

```
.cards {  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
  gap: 1em;  
}  
  
.cards .card {  
  flex-basis: 15em;  
  flex-grow: 1;  
}
```

Когда мы думаем о макетах, мы часто думаем о дизайне на уровне страницы.

Но меньшие компоненты внутри страницы могут иметь свои собственные автономные макеты.

В идеале эти макеты на уровне компонентов будут настраиваться автоматически, независимо от их положения на странице.

Могут возникнуть ситуации, когда неизвестно, будет ли компонент помещен в основную колонку содержимого или боковую панель, или в то и другое вместе.

Не зная наверняка, где в конечном итоге окажется компонент, нужно убедиться, что компонент может адаптироваться к своему контейнеру.

# Микро-макеты


Когда мы думаем о макетах, мы часто думаем о дизайне на уровне страницы.

Но меньшие компоненты внутри страницы могут иметь свои собственные автономные макеты.

В идеале эти макеты на уровне компонентов будут настраиваться автоматически, независимо от их положения на странице.

Могут возникнуть ситуации, когда неизвестно, будет ли компонент помещен в основную колонку содержимого или боковую панель, или в то и другое вместе.

Не зная наверняка, где в конечном итоге окажется компонент, нужно убедиться, что компонент может адаптироваться к своему контейнеру.




**Media Object**

This is a media object. There's an image and also some content.

In this case, the content is a heading followed by two paragraphs of text.

This particular media object is inside the main content.




**Media Object**

This is a media object. There's an image and also some content.

In this case, the content is a heading followed by two paragraphs of text.

This particular media object is inside the main content.



**Media Object**

This is a media object. There's an image and also some content.

In this case, the content is a heading followed by two paragraphs of text.

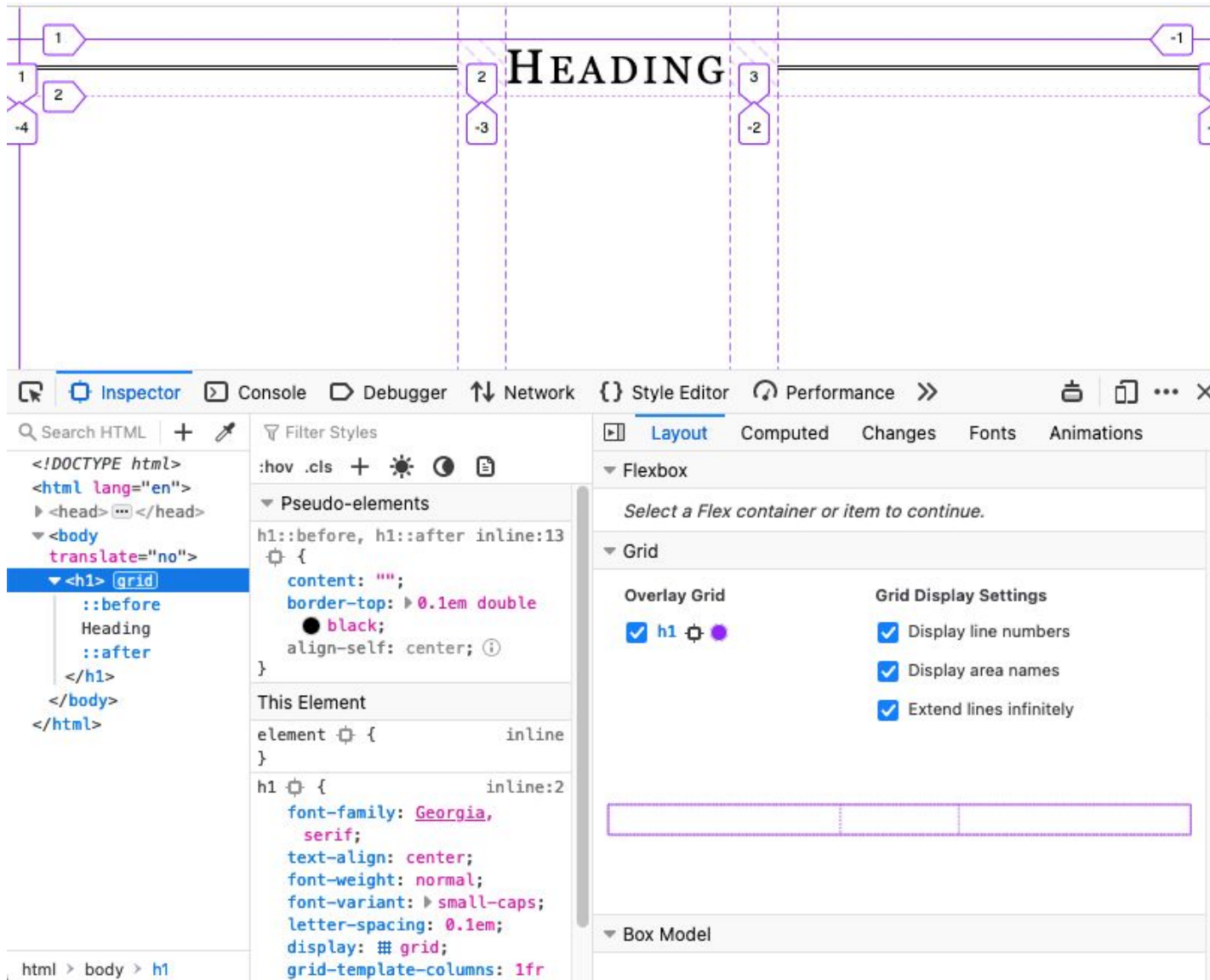
This particular media object is in the sidebar.

Grid предназначен не только для макетов на уровне страницы. Он также хорошо работает для компонентов, которые находятся внутри.

```
h1 {  
  display: grid;  
  grid-template-columns: 1fr auto 1fr;  
  gap: 1em;  
}
```

```
h1::before,  
h1::after {  
  content: "";  
  border-top: 0.1em double black;  
  align-self: center;  
}
```

# Микро-макеты

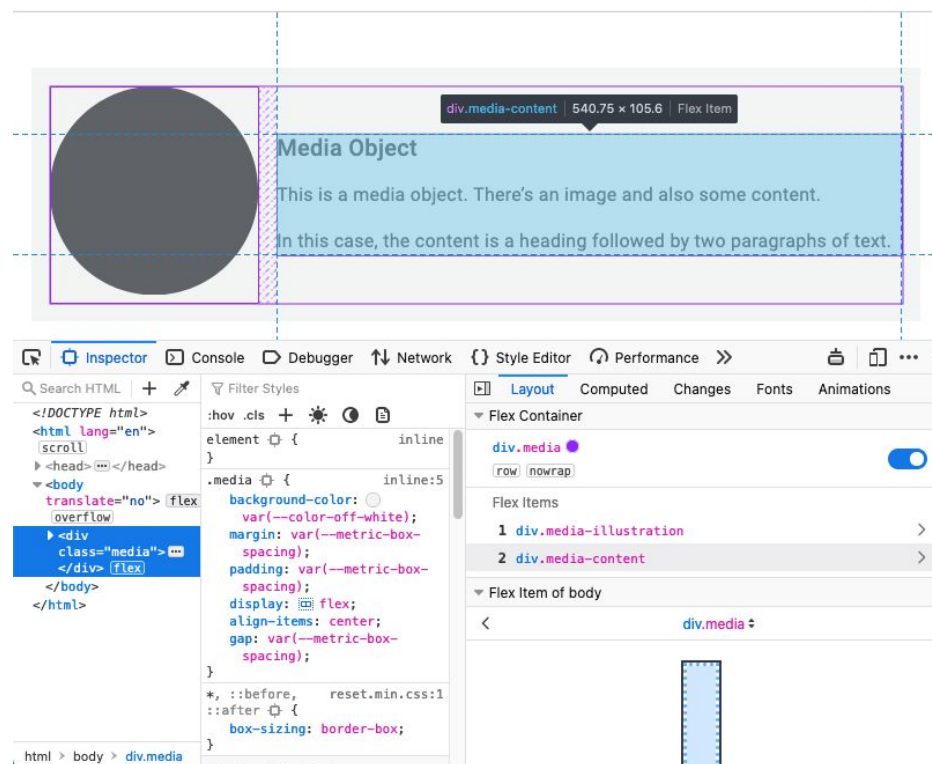


# Микро-макеты

Flexbox позволяет сделать компоненты гибкими. Можно указать, какие элементы в компоненте должны иметь минимальный или максимальный размер, и позволить другим элементам огибать соответствующим образом.

В примере изображение занимает четверть доступного пространства, а текст – остальные три четверти. Но изображение никогда не становится больше 200 пикселей.

```
.media {  
  display: flex;  
  align-items: center;  
  gap: 1em;  
}  
  
.media-illustration {  
  flex: 1;  
  max-inline-size: 200px;  
}  
  
.media-content {  
  flex: 3;  
}
```





Если не указать какие-либо стили для текста, браузеры будут применять свои собственные стили по умолчанию.

Это *таблицами стилей пользовательского агента* и могут отличаться от браузера к браузеру.

Пользователи также могут устанавливать свои собственные предпочтения для отображения текста.

Если не указать длину строки, браузеры будут переносить строки текста по краю экрана. Таким образом, текст в Интернете по умолчанию адаптивный — он перемещается в соответствии с окном просмотра пользователя.

```
@media (min-width: 30em) {  
  html {  
    font-size: 125%;  
  }  
}
```

```
@media (min-width: 40em) {  
  html {  
    font-size: 150%;  
  }  
}
```

```
@media (min-width: 50em) {  
  html {  
    font-size: 175%;  
  }  
}
```

```
@media (min-width: 60em) {  
  html {  
    font-size: 200%;  
  }  
}
```

Используя медиа-запросы для изменения font-size свойства по мере увеличения размера экрана.



Адаптивный подход заключается в том, чтобы позволить ширине устройства пользователя влиять на размер текста.

`vw` – единица измерения в CSS обозначает “ширину окна просмотра”.

Привязка размеров шрифта к ширине окна просмотра означает, что текст будет увеличиваться и уменьшаться пропорционально ширине браузера.

Это затрудняет предсказание размера текста при любой конкретной ширине, но размер текста будет соответствовать ширине браузера пользователя.

Важно, чтобы вы не использовали `vw` само по себе в объявлении размера шрифта, т.к. пользователь не сможет изменить размер текста.

Не верно

```
html {  
  font-size: 2.5vw;  
}
```

Размер текста можно будет изменять, если используются относительные единицы измерения — например, em, rem, или ch.

А функция `calc()` идеально подходит для вычисления

Верно

```
html {  
  font-size: calc(0.75rem + 1.5vw);  
}
```

Пусть браузер сам посчитает.

Это затрудняет точное предсказание размера текста при любой конкретной ширине, но размер текста будет в правильном диапазоне.

Браузер пользователя заботится о том, чтобы точно рассчитать размер текста.

Текст нужно контролировать чтобы он не сжимался и увеличивался до крайности.

Где начинается и заканчивается масштабирование, для этого используется функция CSS `clamp()`

`clamp()` принимает три значения.

1. указывает минимальный размер
2. тоже что передается в `calc()`
3. указывает максимальный размер

```
html {  
  font-size: clamp(1rem, 0.75rem + 1.5vw, 2rem);  
}
```



Задать длину строки непосредственно в CSS нельзя.

Но можно ограничить длину строки через `max-inline-size`

Не устанавливайте длину строк с фиксированной единицей измерения, такой как `px`

Пользователи могут увеличивать и уменьшать размер шрифта, и длина строк должна быть скорректирована соответствующим образом.

Используйте относительную единицу типа `rem` или `ch`

```
article {  
  max-inline-size: 66ch;  
}
```

Более короткие строки текста могут иметь большие line-height значения.

Большие line-height значения для длинных строк текста, ухудшают глазу читателя переход от конца одной строки к началу следующей.

```
article {  
  max-inline-size: 66ch;  
  line-height: 1.65;  
}
```

```
blockquote {  
  max-inline-size: 45ch;  
  line-height: 2;  
}
```

Используйте безразмерные значения для line-height объявлений.  
Это гарантирует, что высота строки будет относительно font-size

```
line-height: 1.5;
```



Передовые  
инженерные  
школы



МИНОБРНАУКИ  
РОССИИ



УНИВЕРСИТЕТ  
ИННОПОЛИС

# Спасибо за внимание

