Модуль А3.

Основы типографики – что это.

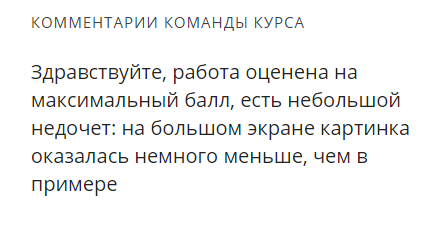
Подключения шрифтов к сайту, локальные и **CDN.**

Эффективная верстка быстро и адаптивно.

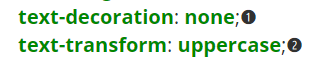
Адаптивная верстка под разные устройства.

**Сначала разбор прошлой практики.**

Меня за нее похвалили.



Сохранил код их примера, ничего особенного. Но есть интересные строки:



Убрать декорацию текста, например, подчеркивание для линий.

Сделать все буквы в тексте в верхнем регистре.



Вертикальное выравнивание по верху.



Прозрачность текста 70%

**Типы шрифтов.**

В вебе есть несколько типов шрифтов, их называют **семействами.**

**Serif –** шрифты с засечками.

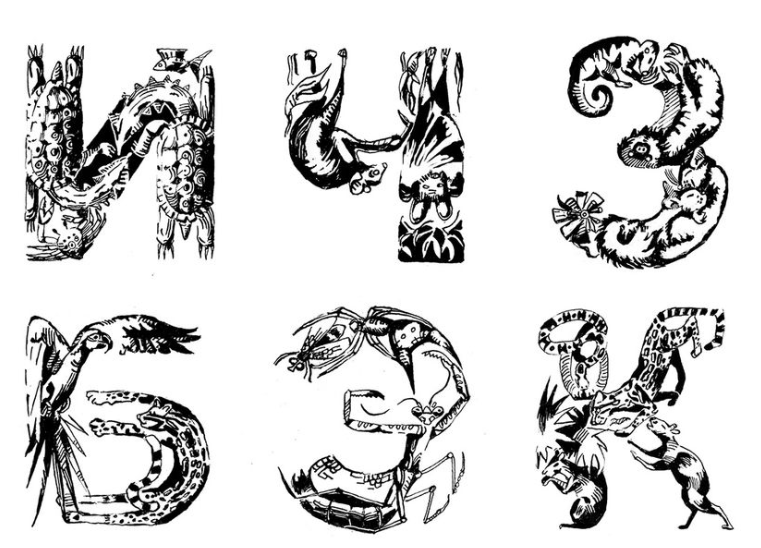


**Sans-serif –** шрифты без засечек.



Сайта MDN к примеру, для заголовков использует шрифт с засечками.

**Fantasy** - декоративные шрифты



Для отдельных надписей и лого.

**Monospace** - моноширинные шрифты



символы в таких шрифтах занимают одну ширину. Чаще всего используются в редакторах кода.

**Cursive** - рукописные шрифты



Имитирует рукописный текст. Иногда состоит не из букв, а иконок. Часто используется **Font Awesome.** [**https://fontawesome.ru/all-icons/**](https://fontawesome.ru/all-icons/)

Можно и свои шрифты создавать. Например через [Glyphter](https://glyphter.com/) - <https://glyphter.com/>

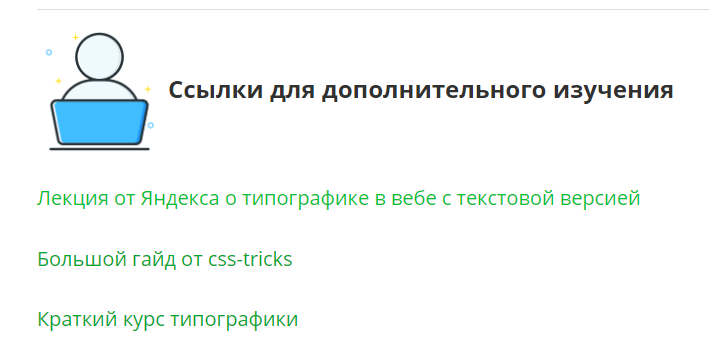
Подключать иконочные шрифты, целесообразно только если иконок много. Ради пары иконок подключать целый шрифт не выгодно, лучше сделать это через изображения.

**Больше о шрифтах**. <http://www.quickandlazy.appspot.com/>

Как подключать шрифт?



В браузерах шрифты по умолчанию могут отличаться, кроме того их можно настроить.

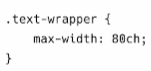


<https://habr.com/ru/company/yandex/blog/432560/> **+**

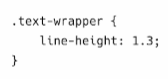
<https://css-tricks.com/typography-for-developers/> **+**

<http://www.quickandlazy.appspot.com/> **+**

В тексте лучше иметь не более 10 слов в строке, это около 80 символов на строку.



Должно быть удобное межстрочное расстояние.



Ну и удобный размер шрифта



Это не везде работает, но можно задать свойство hyphens: auto; Для автоматических переносов строки.



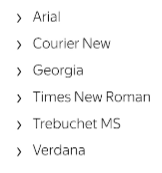
Текст не плохо бы прогнать через типограф, он исправит многие косяки, например, маленькие дефизы.

**Variable fonts –** интересная технология для шрифтов.

Системные шрифты – по умолчанию.

Установленные – думаю понятно.

**Безопасные шрифты**



Функция local – может проверить присутствует ли шрифт в ОС пользователя.

У шрифтов можно выбирать раскладку, тем самым подключая например только латинские буквы или только кириллицу, тем самым не загружая сайт необходимость загружать не использованные символы.



Загрузка шрифтов может быть сделана с помощью javascript.

FOUT – Flash of unstyled Text. Вспышка не стилизованного текста.

Можно отрендерить прозграчным шрифтов или блочном

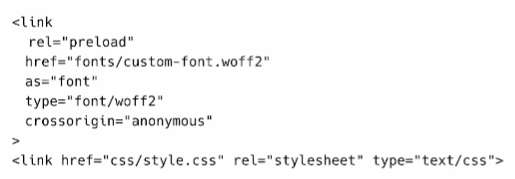


Но в основном используется block или swap.

**block** заставит подождать сайт, 3 секунды и только затем перейдет к следующему прописанному шрифту. А **swap** выберет следующий шрифт сразу.

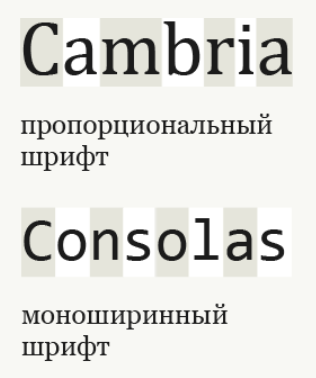
А вообще почему шрифты могут грузится долго? Дело в том, что сначала загружается html, затем к нему подключаются стили и только когда эти стили запарсятся начнется загрузка шрифтов из этих стилей.

Но можно сразу загрузить шрифты еще на этапе html это называется **подгрузка шрифтов – preload**



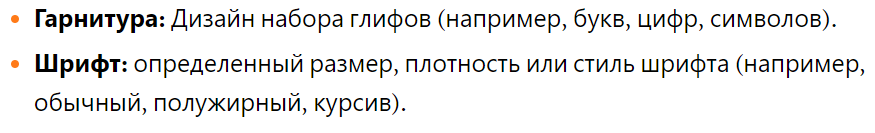
Буква **i** и буква **m** очевидно занимают разную ширину. Это отражается в типах шрифтов. Пропорциональные шрифты – шрифты, в которых ширина символа рознится в зависимости от символа.

Моноширинные – шрифты, где ширина каждого символа одинаковая.



Наклонный шрифт и курсив это не одно и тоже.

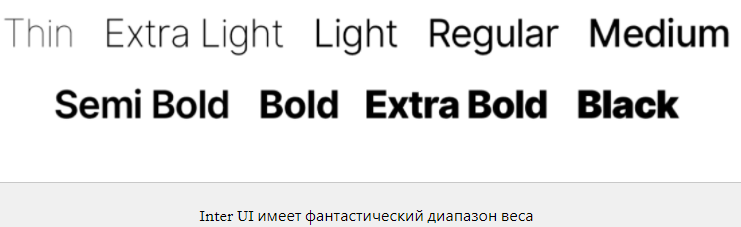
Наконец понял разницу между **гарнитурой** и **шрифтом.** Гарнитура — это сама форма написания цифр, букв и символов. А шрифт, это уже различные формы этой гарнитуры, жирные, курсивные, обычные в целом.



**Гарнитура – Typeface.**

**Шрифт – Font.**

**Некоторые шрифты имеют широкий диапазон толщины.**



**Лигатура –** в некоторых шрифтах, обычно с засечками есть глифы для замены неудобных символов. которые представляют собой несколько символов в одном глифе.

**Пример:** ﬃ и ﬄ

Шрифты сильно влияют на производительность сайтов, рекомендуется их сжимать, например, через GZIP. Отключать не нужные функции - hinting, kerning. Удалить не нужные глифы, например символы иностранных языков.

**Относительно размеров шрифтов.**

Бывают абсолютные размеры и относительные. Будем рассматривать оба на примере двух вариантов (их больше чем два). **PX** и **EM.**

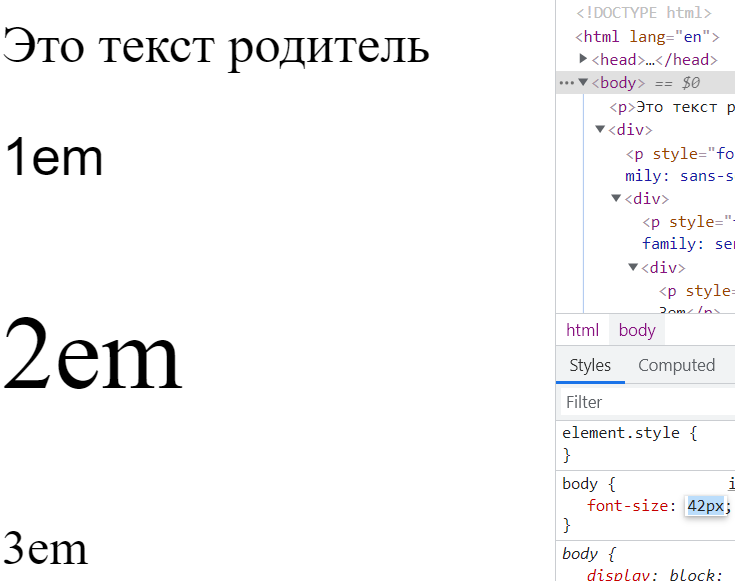
**px абсолютная величина.**

Условные 16px у одного шрифта, могут быть заметно больше или меньше тех же 16 px у другого шрифта.

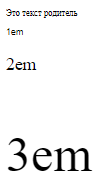
**EM относительная единица.**

Текст размером 2em будет в двое больше размера шрифта его родителя.

**Пример с body**



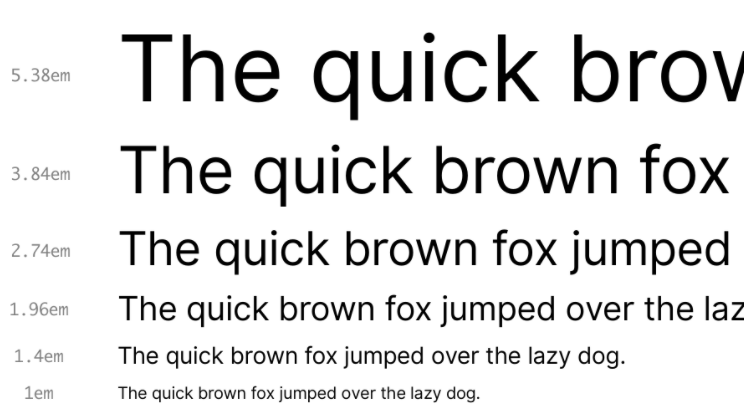
Для body задаю стиль размер шрифта. И этому размеру будет соответствовать 1em. Надписи 2em задан стиль 2em, поэтому она всегда будет в два раза больше стиля 1em. Надписи 3em задан стиль font-size: 40px. И какие бы изменения не происходили с font-size в body, надпись 3em останется 40px.



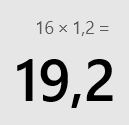
Пример применения модульной иерархии размера em с коэффициентом 1.4

То есть размер с каждым разом будет умножаться на 1.4

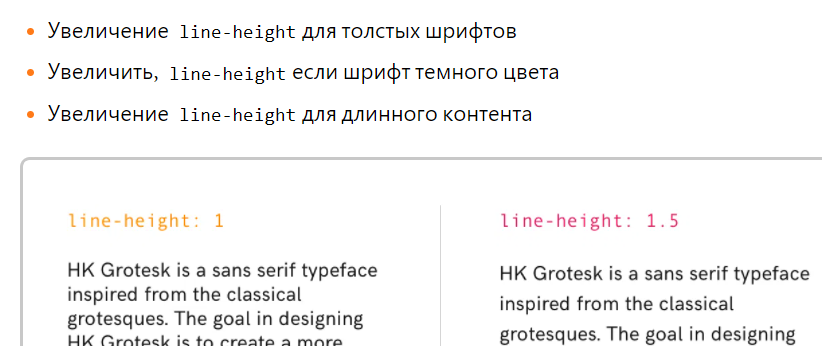
1 1\*1.4= 1.4 1.4\*1.4 = 1.96 и тд.



Также размер шрифта будет изменять от размера строки **line-height.** Допустим задан шрифт 16px, но высота строки задана 1.2. То таким образом высота каждой строки будет равна 19.2

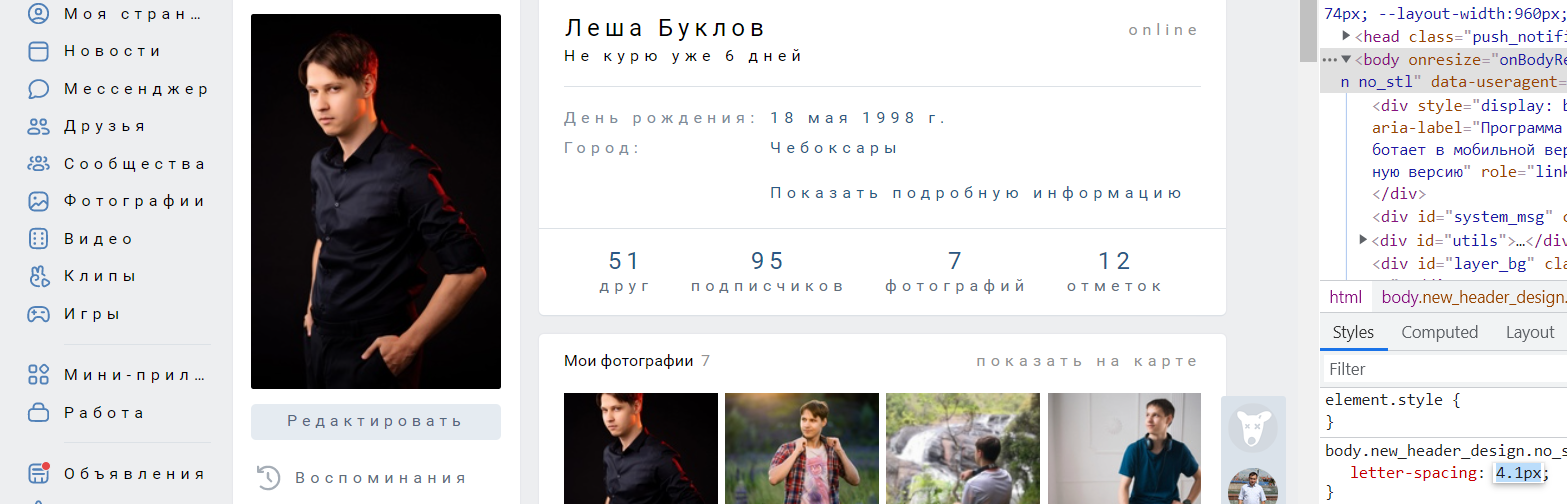


Поэтому не стоит забывать про высоту шрифта. Вот некоторые рекомендация по этому поводу.



Но также важно и сохранять расстояние между символами. Это можно регулировать свойством letter-spacing:

по умолчанию оно равно нулю кажется, но если его увеличить…



То оно увеличится. Его вполне можно регулировать для особенно толстых шрифтов. Но только по чуть-чуть. По 0.01 к примеру.

**Подключение шрифтов**

Шрифты представляют из себя файлы форматов ttf, woff, eot  и т.д.

Отображаться он будет если он установлен в ОС, или подлечен в коде сайта.

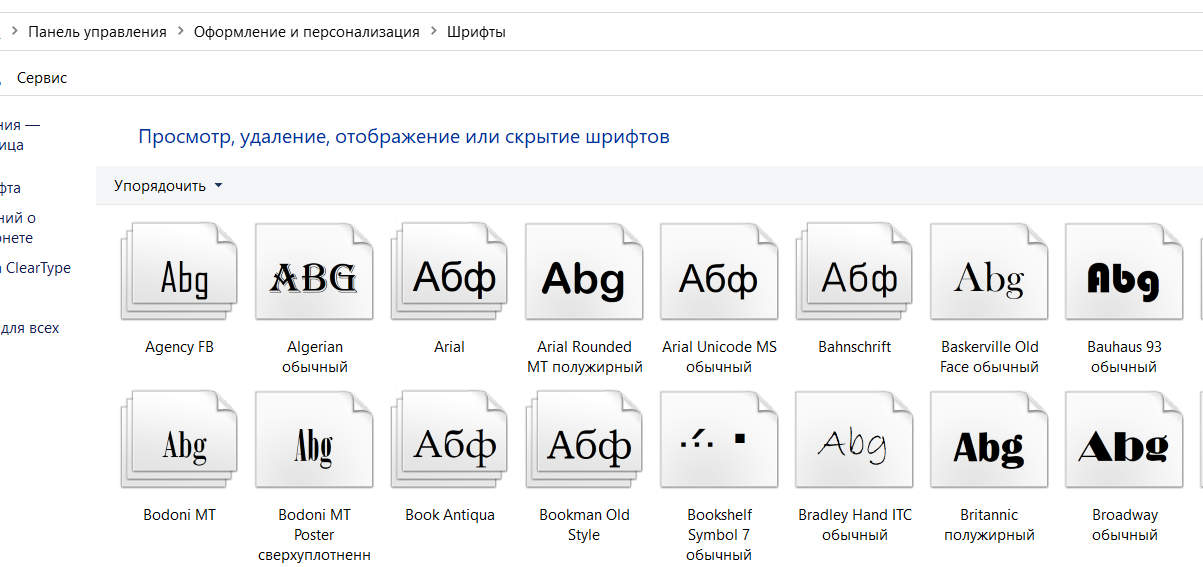
Подключать их можно по-разному.

**Локальное подключение:**

шрифт лежит на том же сервере, что и сайт.

Нужно скачать файл со шрифтом, бывают как платные, так и бесплатные, нужно читать лицензии.

Установленные шрифты:



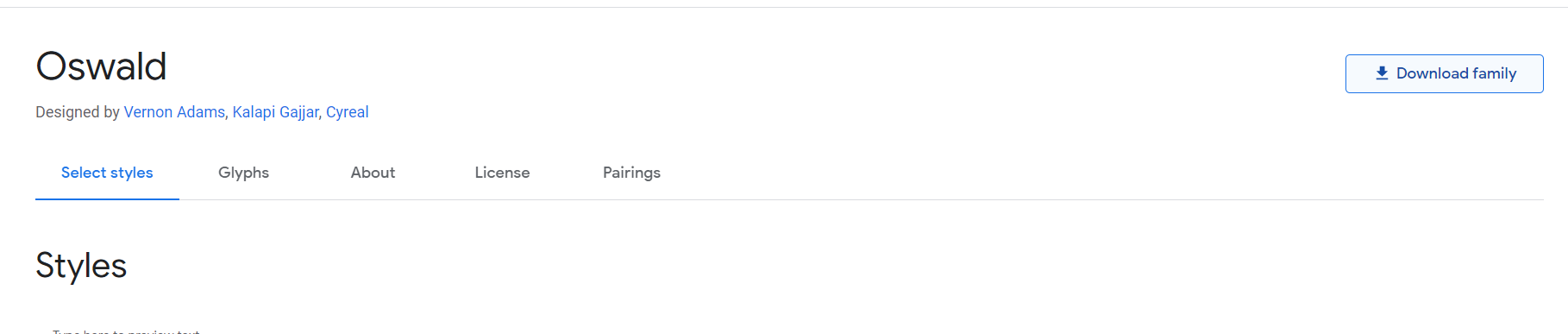
Шрифт скачивается архивом с несколькими файлами, файл с лицензиями тоже там. В проект не обязательно подключать все файлы, а только те, что используются.

Подбирать шрифты можно на многих сайтах. <https://fonts.google.com/>

Тут нужно нажать на окно с шрифтом



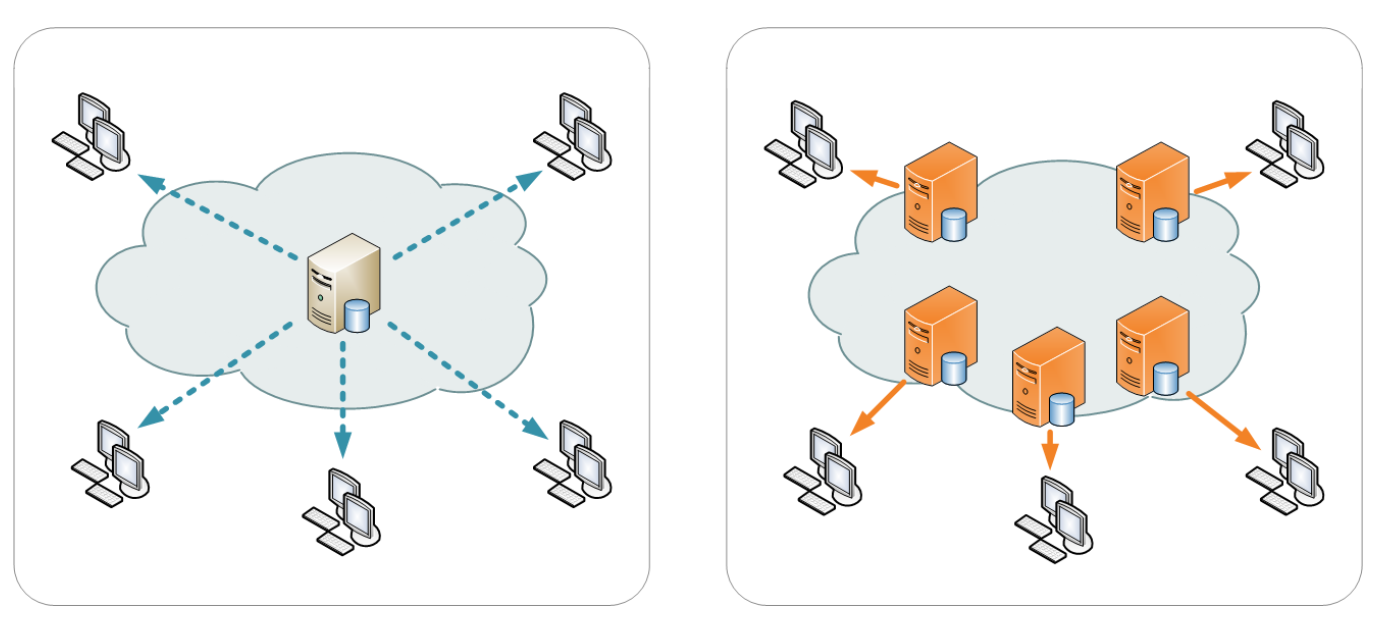
И затем нажать на кнопку скачать.



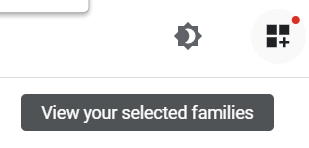
Проблема локальных шрифтов в том, что чем дальше пользователь находится от сервера, тем дольше файлы будут загружаться. Для решения этой проблемы используется **CDN(Content Delivery Network) –** сеть доставки контента.

Эта сеть расположена по всему миру, она отдает ресурсы в зависимости от местоположения клиента.

Вот наглядная разница между сервером и CDN.



Многие сервисы загружают шрифты в CDN. И вместо скачивания на компьютер можно просто подключить ссылку на стиль, это оптимизирует время загрузки файлов.



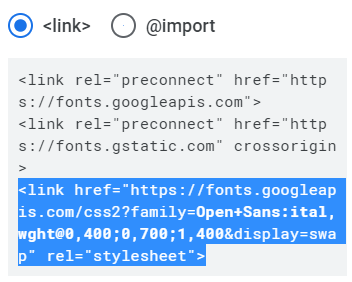
**Кнопка с точкой для просмотра выбранных стилей**

Подключение схоже со стилями, через тег <link>. Способ через html.



Через link подключаются не только стили и шрифты, но и другие ресурсы.

Выбирая ссылки на том сайте, нужно выбирать нижний вариант:

****

**Подключение шрифтов в CSS**

В CSS тоже есть несколько способов подключить шрифты.

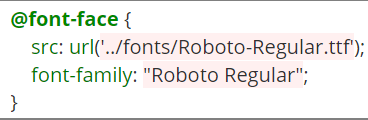
С помощью правила **@import** указать ссылку на файл шрифтов (на сервере или CDN). Делать это нужно в начале CSS файла. Или внутри тега **style** внутри head html.

**Пример:**



Второй способ более гибкий. Через **CSS** правило **@font-face.** В нем указывается путь до шрифта и его название.

**Пример:**

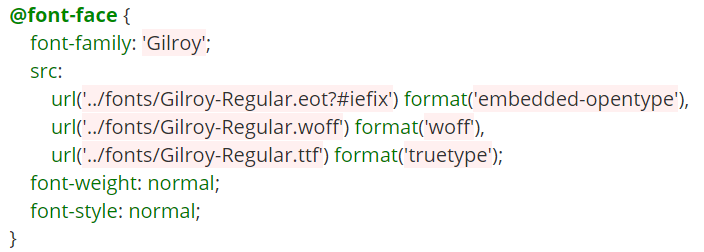


Все css правила начинаются с “собаки” **@**

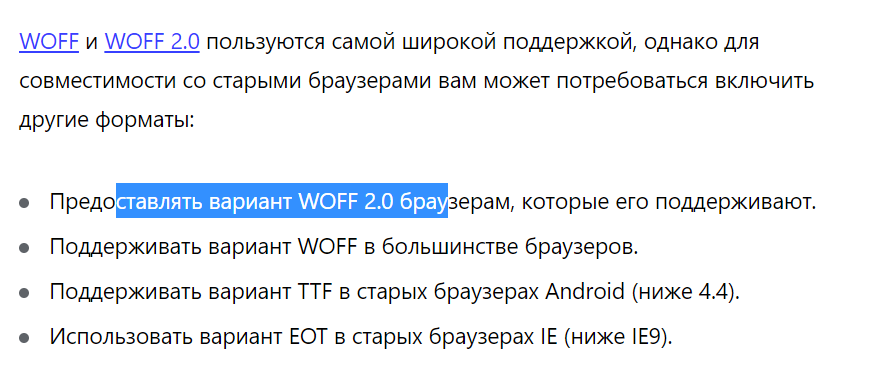
**src** указывает путь к файлу шрифтов. Путь может быть относительным или абсолютным.

Не существует единого формата шрифтов, который бы одинаково хорошо читался во всех браузерах. Поэтому для максимального покрытия всех браузеров лучше подключить 3-4 разных формата шрифтов.

**Пример** как подключить несколько шрифтов.



В этом примере присутствует директива **format()**, которая подсказывает браузеру, какой формат шрифта используется.



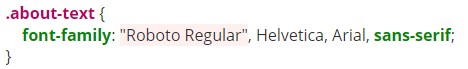
О том, как оптимизировать сайт можно прочесть здесь. <https://web.dev/fast/#optimize-webfonts>

В свойстве font-family можно указать любое название, если в названии больше одного слова, то обязательно завернуть в двойные кавычки.

Для каждого начертания (жирный, полужирный, курсив) используется свое правило, т.к. каждое начертание — это отдельный файл.

Подключенный стиль нужно включить.

**Пример:**



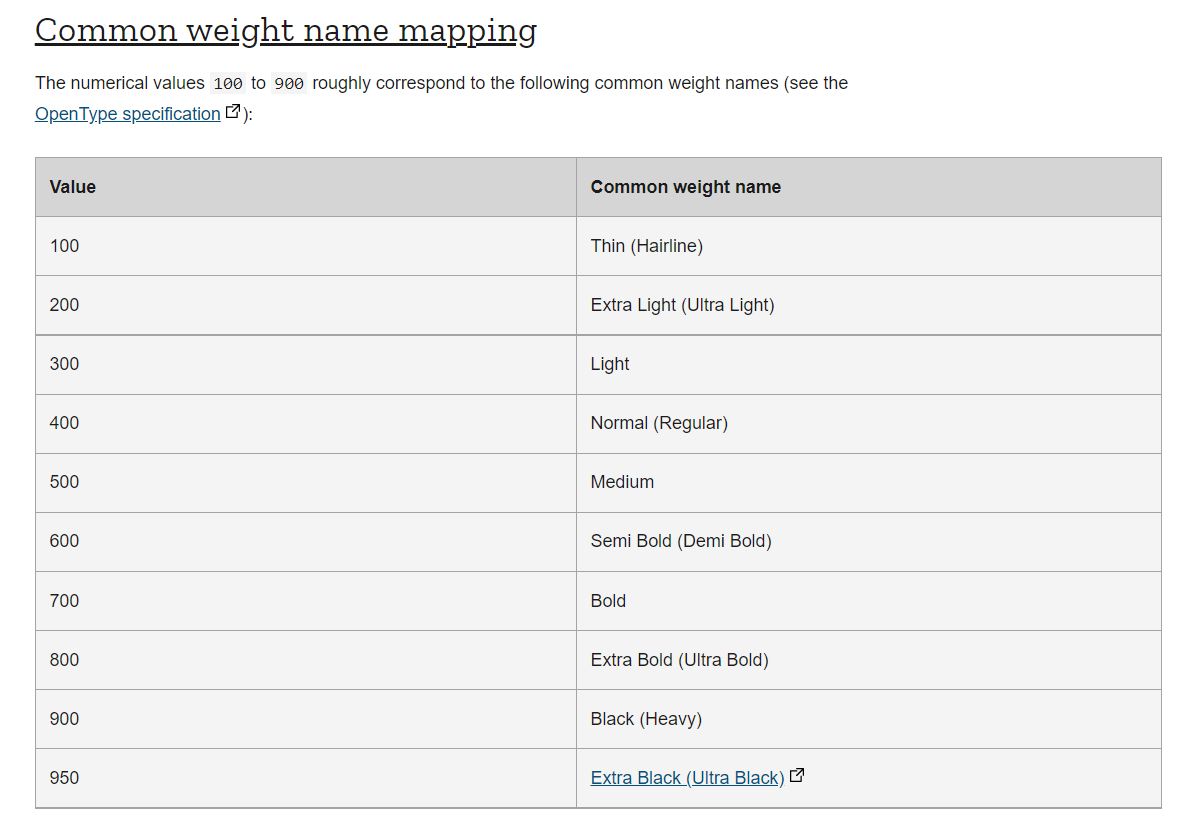
Если шрифт не подгрузился из-за каких-либо неполадок, то браузер будет использовать шрифты, указанные после, запасные.

Если и они не могут быть открыты, то браузер использует шрифт, который прописывается в ОС и браузере. В windows в качестве шрифта **sans-serif** можно использовать **Areal.** Популярные шрифты по умолчанию можно посмотреть тут: <https://www.cssfontstack.com/>

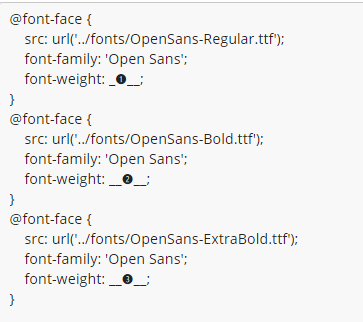
Шрифты выбираются слева на право, сначала основной, затем запасные, а затем семейства.

Запутала меня эта практика.

Узнать нужный вес можно при выборе начертаний на сайте Google Fonts:

[https://developer.mozilla.org/en-US/docs/Web/CSS/font-weight#common\_weight\_name\_mapping](https://developer.mozilla.org/en-US/docs/Web/CSS/font-weight#common_weight_name_mapping)

Как указать вес:



**Основы флексбокса.**

Строчные и блочные элементы для раскладки страницы устарели. Эффективнее верстать можно с помощью более современных способов. Один из них Флексбокс.

Раскладка флексбокс позволяет быстро расположить элементы на странице в соответствии с дизайном макета. Флексбокс – более гибка альтернатива **inline-block.**

**Как он устроен?**

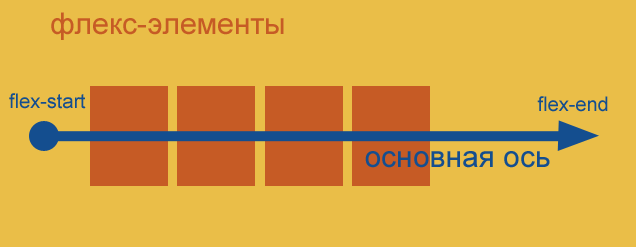
Ранее мы задавали свойство **display** каждому блоку, который нужно было куда-то расположить.

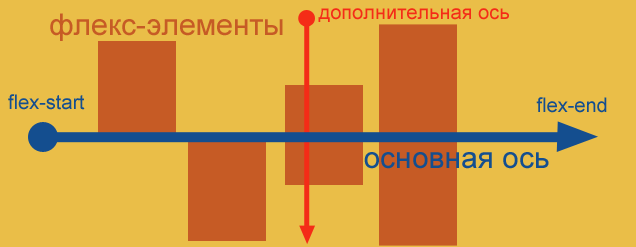
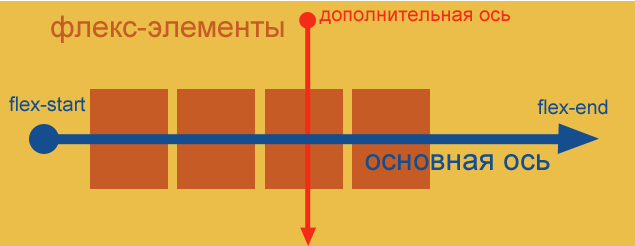
С флексбоксом сначала создаем обертку – родителя или контейнер. Этому элементу задаем свойство **display: flex.**

Непосредственные потомки этого контейнера станут флекс элементами. Их расположением и порядком можно будет управлять.

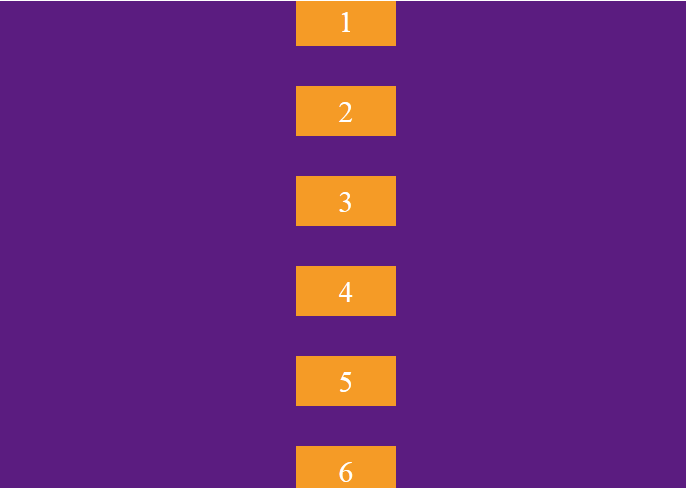
В роли контейнера часто выступает элемент **<DIV>.** Просто обертка. Но контейнером может выступать любой элемент, например, список **<ul>**

Флекс элементы выстраиваются внутри контейнера вдоль главной оси, по умолчанию слева на право, это можно будет изменить:



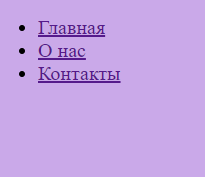
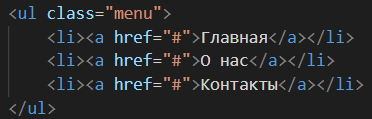


Дополнительная ось перпендикулярна основной. По умолчанию проходит сверху вниз.

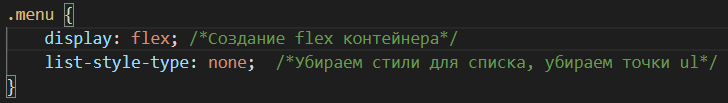


Попробуем применить это для примера.

Создадим список с несколькими элементами внутри для шапки.



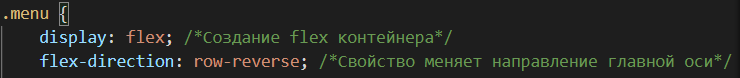
Если прописать в стилях **flex.** То, элементы встанут по главной оси.





Это направление можно будет поменять, тогда список окажется не в левой части сайта, а в правой.

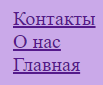
Для этого используется свойство **flex-direction.** По умолчанию значение **row,** но можно изменить на **row-reverse.**



И также меняются значение второй оси:

**column –** сверху вниз.

**column-reverse –** снизу вверх



Игра flex-froggy. <http://flexboxfroggy.com/#ru> Для практики. **+**

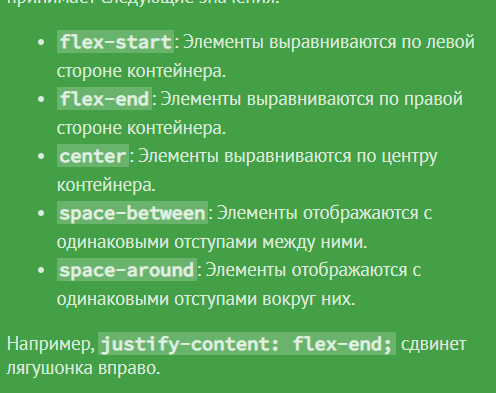
Еще пару ссылок

<https://bureau.ru/bb/soviet/20170914/> - Интерактивная демонстрация свойств флексбокса **+**

<https://www.freecodecamp.org/news/an-animated-guide-to-flexbox-d280cf6afc35/> - **+** Как работает флексбокс в гифках

Свойства из игры про лягушонка:

**justify-content;** для расположения



Есть путаница между **space-around** и **space-between.**

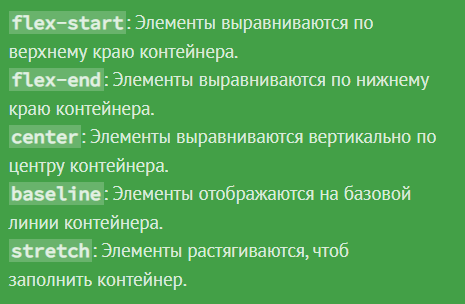
**Space-around –** одинаковые отступы вокруг элементов.



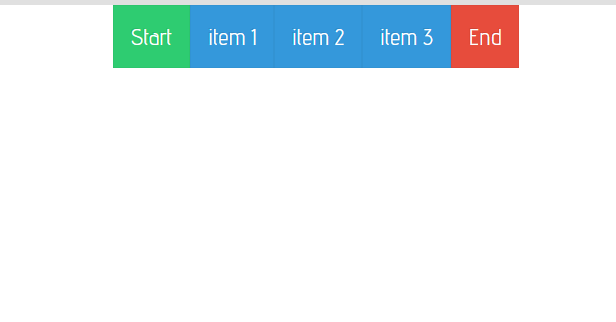
**Space-between –** одинаковые отступы именно между элементами.



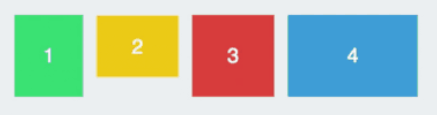
Теперь выравнивание по вертикали. **Align-items:**



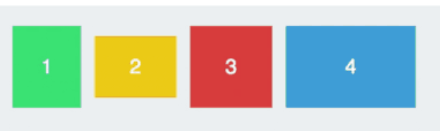
**baseline(**слегка отличается от **flex-start,** разница если элементы разной высоты**)** и **stretch**



**Flex-start**



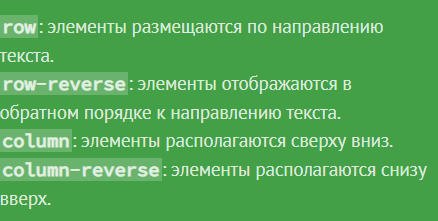
Например, если есть тег абзаца, то элементы станут так:



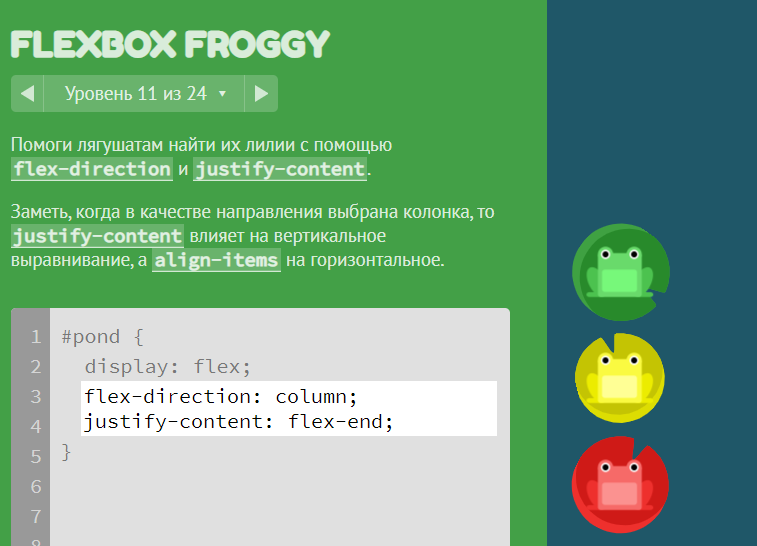
А без него так:



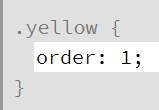
**Направление flex-direction:**



Если мы меняем **flex-direction** на **column**, то свойства **justify-content** и **align-items**, меняются местами.

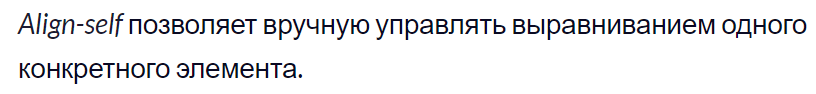
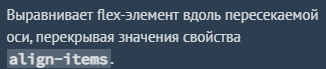


Иногда размещать так элементы недостаточно и, например, нужно поменять два соседних элемента местами. Для этого есть свойство **order,** по умолчанию оно равно 0.

У зеленого и красного оно ноль, а у желтого 1. Поэтому он встает правее.

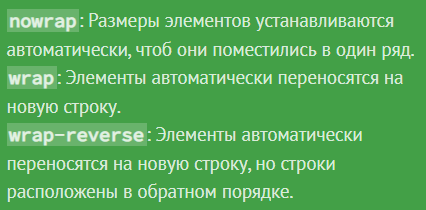
Свойство **align-self:** принимает те же значения, что и **align-items.** Но разница в том, что оно применимо, только для **одного элемента. ??? Я не знаю, а в чем разница.**



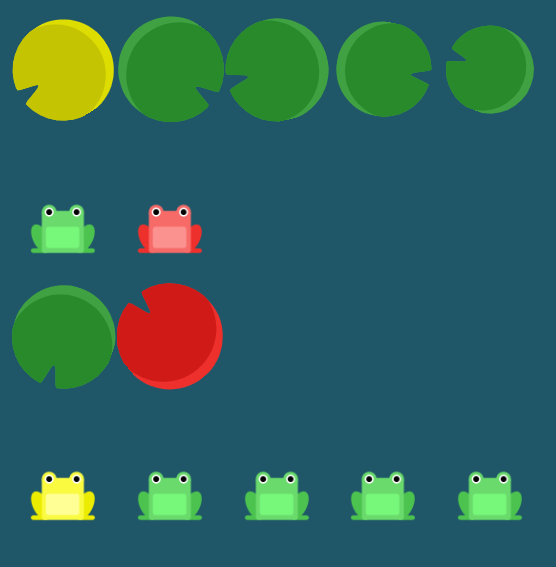
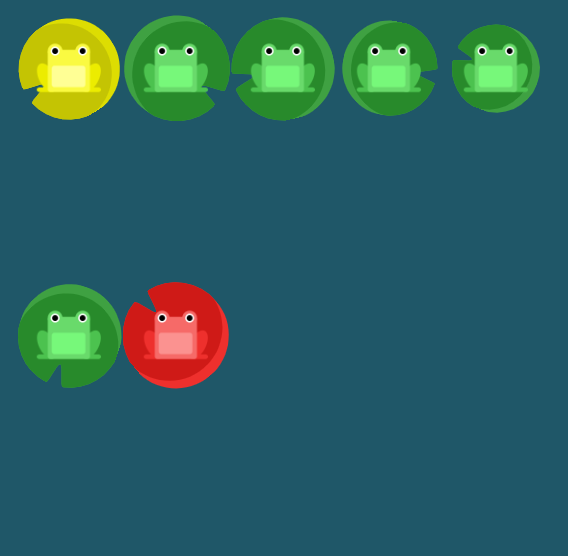
**У него приоритет больше.**

Элементы могут быть сплющены из-за своего количества. Их можно растянуть с помощью свойства **flex-wrap. Отвечает за перенос на новую строку.**





wrap-reverse и wrap

Свойства **flex-direction** и **flex-wrap** используются вместе очень часто. Поэтому для них создали объединённое свойство. **Flex-flow.**



Помимо **justify-content** есть еще **align-content.** Значения свойств такие же.

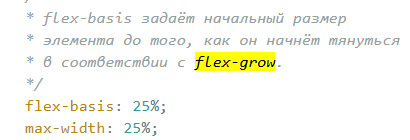
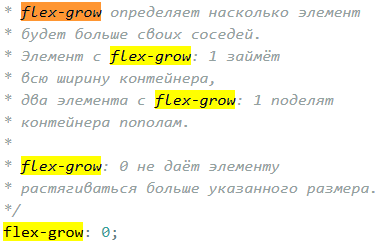
Подходит для вертикального перемещения нескольких рядов.



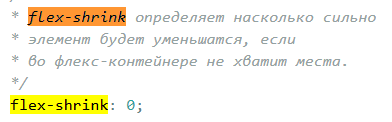
Прошел. Похожее будет позже по гриду.

Еще пару свойств, которые не были упомянуты.

**Flex-grow:**  **Flex-basic**



**Flex-shrink**



Еще пару ссылок по флексам.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/> <https://developer.mozilla.org/ru/docs/Learn/CSS/CSS_layout/Flexbox>

<https://www.freecodecamp.org/news/understanding-flexbox-everything-you-need-to-know-b4013d4dc9af/>

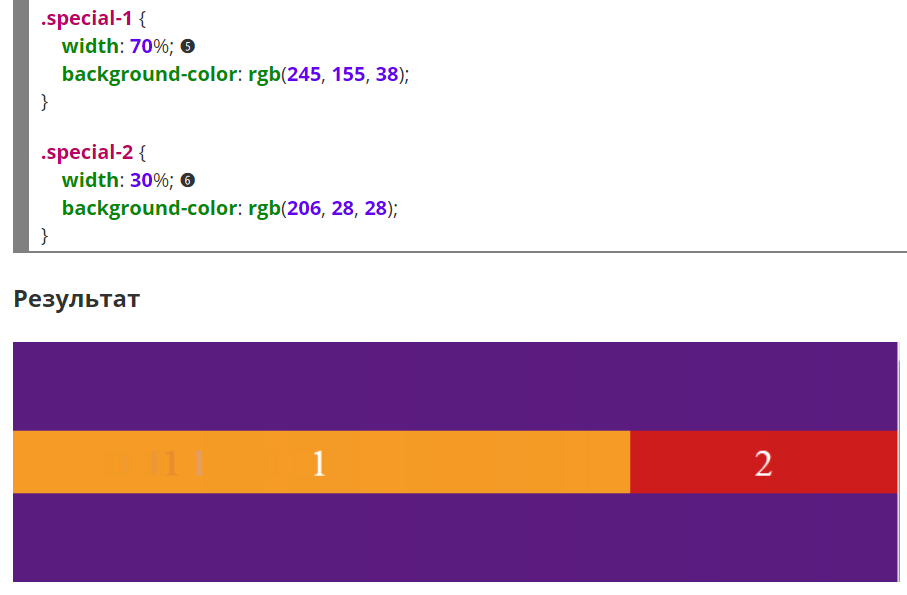
<https://www.freecodecamp.org/news/the-ultimate-guide-to-flexbox-learning-through-examples-8c90248d4676/>

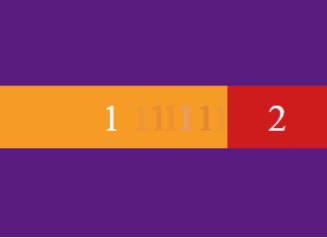
**Расстояние и размеры**

Размеры блокам уже задавать умеем, с помощью пикселей и процентов. Научимся работать и с другими. И заодно применим их в связке с флексбоксом.

Элементам можно задавать **относительные значения.** В таком случае их ширина будет меняться вместе с шириной экрана или шириной родителя. Такая верстка называется **резиновой.**

Можно отключить фиксированную ширину для блоков и каждому блоку задать ширину в процентах, в таком случае эти блоки при любой ширине экрана будут иметь такое процентное соотношение. с





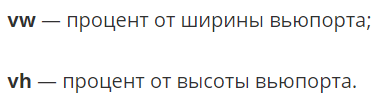
В случаях, когда нет уверенности в том, сколько элемент будет занимать места можно задать ему минимальную и максимальную ширину и высоту.

Если задать их, то элемент будет ужиматься до определенного момента, а затем будет исчезать.



Когда вы используете **проценты**, размер элемента рассчитывается в зависимости от ширины **родительского элемента**. Это может быть неудобно, если родитель занимает не всю ширину, а вы хотите, чтобы элемент занял половину экрана.

Если вместо процента использовать единицы **vw** и **vh,** то процент будет высчитываться не от родителя, а от вьювпорта.



Граница красной зоны, это ровно половина от ширины экрана, а не половина от фиолетовой области. (Фиолетовая зона не затрагивает весь экран, а только 90 процентов).

**Это полезная информация, ее лучше держать в голове.**

**Отступы**

Помимо **фиксированных отступов** флексбокс позволяет задавать автоматические. Они распределяют все свободное пространство.

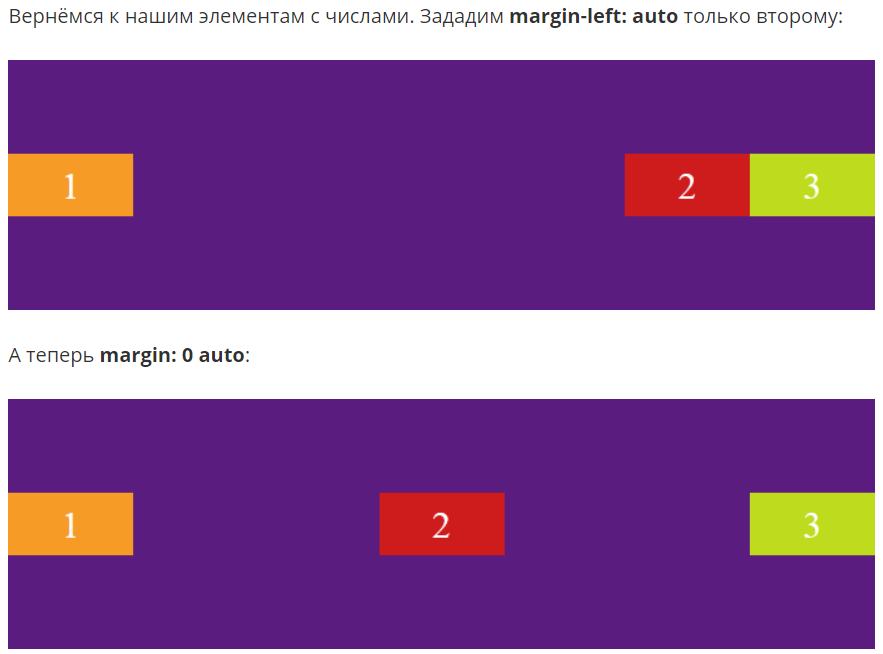
**Пример:**



Если задать этим блокам отступ слева автоматический: **margin-left: auto;** То отступ слева займет все доступное место и элемент прижмется к правому краю.



Вообще-то автоматический внешний отступ работает и без флекс-боксов. Любой блочный бокс можно так выровнять. Часто это делают, чтобы расположить элемент по центру страница, например, шапку. Для этого нужно отступы слева и справа указать auto: **margin: 0 auto;**



Единицы измерения - <https://learn.javascript.ru/css-units>

**Верстка Pixel Perfect**

Верстка пиксель в пиксель нужна не всегда. Да и такой она будет только для определенной ширины экрана. Для упрощения такой верстки есть расширения для браузера. Например **PixelPerfect** для Chrome, которое позволяет наложить jpg изображение с макетом поверх страницы с определенной прозрачностью и подгонять все расстояния.

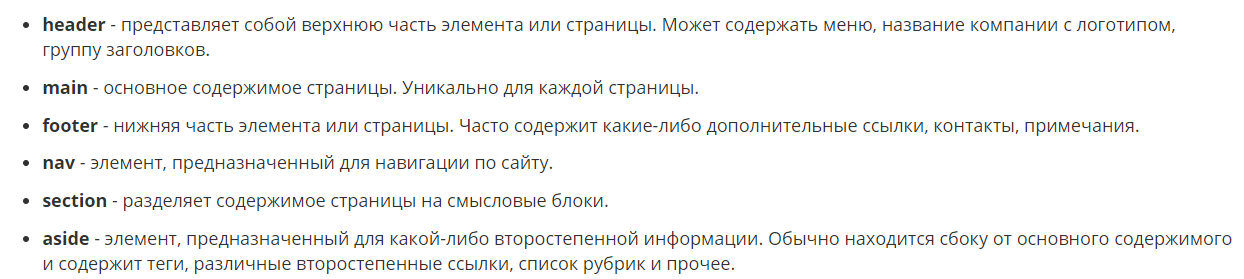
Важно, чтобы это расширение работало не только на реальных сайтах, но и локальных файлах на самом компьютере нужно в настройках расширения установить эту галочку.

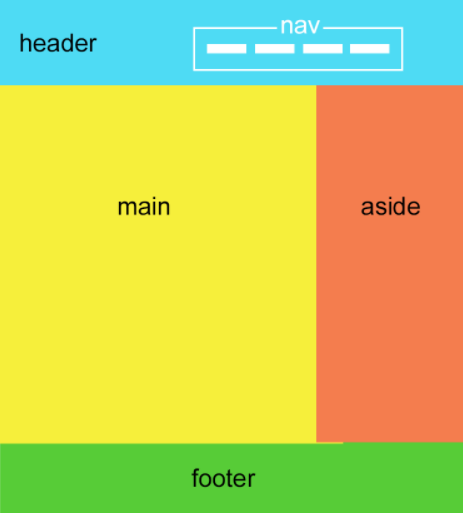


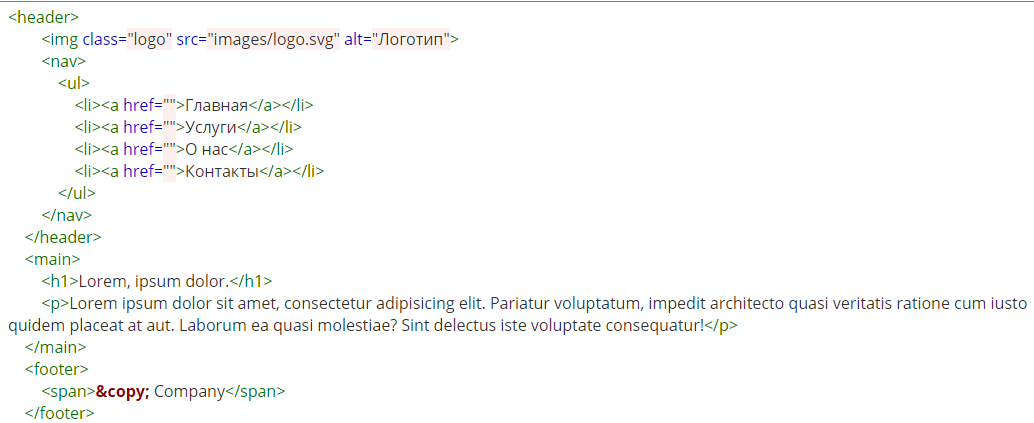
Но этим нужно заниматься в конце. Не в начале.

**Семантические элементы**

Уже со многими встречался. header, main, footer, nav, section, aside.







Сейчас сверстаю старый макет с новым знаниями. Конкретно с Флексами. Но конструкция flex header -> ul -> li -> a не самая лучшая, т.к. header становится контейнером, а ul flex блоком, но li и a, не являются flex блоками, чтобы правильно расположить все li , нужно ul задать display: flex тоже.

Нет, кажется я еще не готов.

**Основы адаптивной вёрстки**

Адаптивная верстка – верстка, адаптированная под разную ширину экрана. Отличается от резиновой тем, что не просто растягивает элемент, но и меняет раскладку.

Можно и совмещать с резиновой версткой задавая блокам максимальную и минимальную ширину, но суть в том, что перенос на новую строку должен быть. А значит должно быть свойство: **flex-wrap: wrap;**

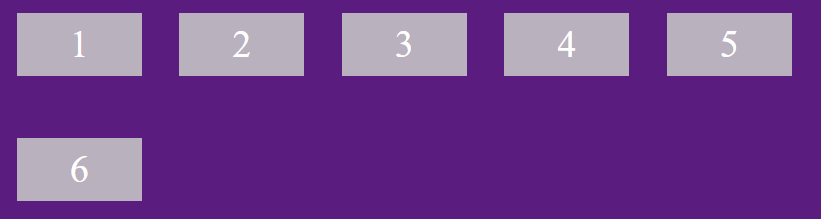
Однако продумать верстку под любую ширину почти невозможно. Но можно сделать ручную подгонку с помощью CSS правил, или если конкретнее **медиа запросов.**

@import, @font-face и т.д.

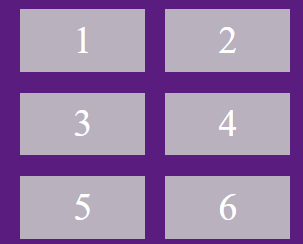
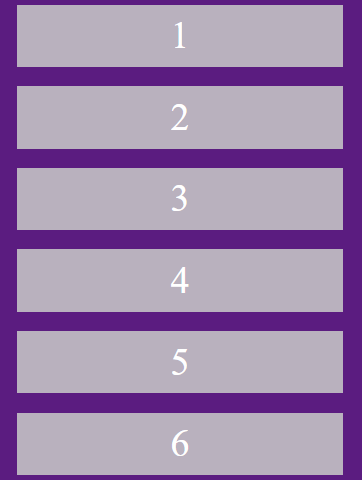
Медиа-запрос позволяет изменять, удалять и создавать новые CSS-правила для определённых размеров экрана.

Что же это действительно любопытно, интересно какие еще формы условий будут доступны, но вот не большой пример.

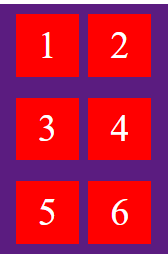
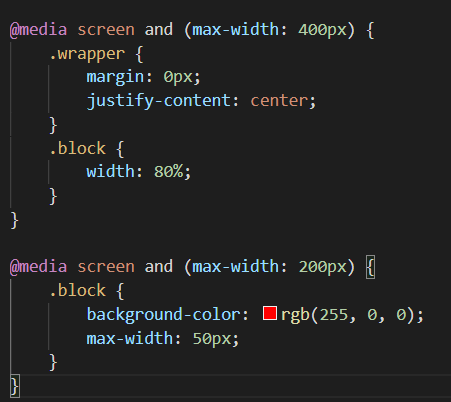
**Элементы в ряд.**



Ужимаем их до 400 пикселей. А затем ниже 400 и срабатало правило и они теперь стали в центр

Есть также дополнительное правило, ниже 200. Оно меняет цвета блоков и максимальную ширину пикселей. А сами свойства описаны так:

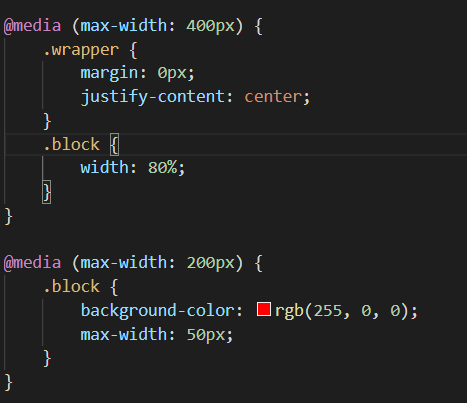
@media – медиа запрос

screen – похоже свойство относящееся к экрану

and () – и само условие срабатывания полагаю?

Внутри запроса уже описаны новые css правила.

Вообще написано, что в большинстве случаев screen можно и опустить и это странно. Эти 2 правила работают и так:

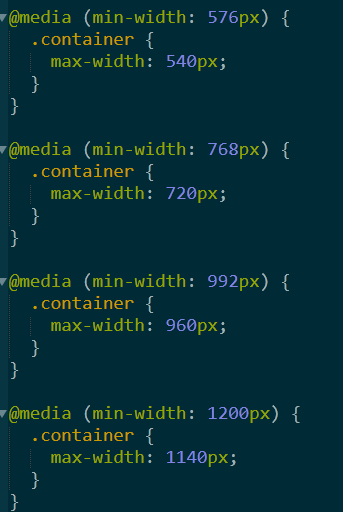


После @media мы указываем, о каком устройстве речь. Как правило screen. В большинстве случаев screen можно опустить, но не всегда. После screen оператор **and** обязателен. Выбирая ширину мы выбираем ширину вьюпорта, а выбирая min или max мы выбираем диапозон, от или до какой ширины будет применяться свойство.

Изменяя свойство нужно писать только те свойства, которые изменяем, не нужно копировать предыдущие свойства.

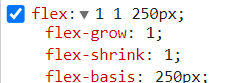
Таким образом можно создавать несколько точек, в промежутках между которыми будут действовать определенные свойства. Их также называют **брейкпоинтами.**

Получая что-то вроде таких наборов правил:



Больше о медиа запросах: <https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries>

У flex объекта есть такое свойство: flex. На самом деле, оно содержит несколько свойство, в таком порядке: **flex-grow, flex-shrink, flex-basic**





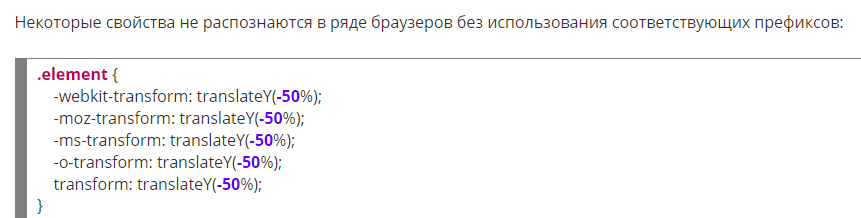
**то можно воспользоваться Github Pages.**

**Тестировать сайт необходимо на множестве экранов и на мобильных устройствах тоже.**

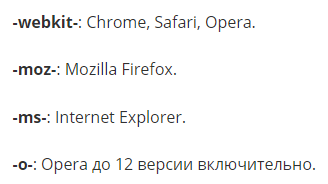
Но большинство сервисов позволяют это протестировать только, запустив локальный сервер.

Также тесты на производительность и слабый интернет.

Даже тестирование на разных браузерах вызывает сложности. Если html везде одинаковый, то css в браузерах может рознится. Например, какие-то свойства могут работать только на определенных версиях браузера, некоторые работают, но показывают разный результат, а некоторые не работают без специальных префиксов.



Каждый префикс относится к тому или иному браузеру.



Чтобы понять, нужно использовать префиксы или нет, можно обратиться к сайту <https://caniuse.com/>

4.11 сначала я хочу сверстать макет, а уже потом смотреть, как надо. Включу следующий вебинар.

Сделал практику, потратил много времени из-за лени. Надо поднапрячься. Посмотреть пару вебинаров и продолжить.

**Вебинар сложные места в модуле А3.**

Первое, эти отступы по бокам измерять не надо. Они будут зависеть от ширины экрана.

Нужно задавать максимальную ширину самой секции и автоматические отсутпы



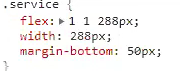
Адаптивность изначально должна быть такая, чтобы ничего не вылезало за край вьюпорта, остальное уже позже дорабатывать, это в приоритете.

Для того, чтобы в адаптивности изменять размер элемента нужно в медиазапросах к определенной ширину задавать ширину.



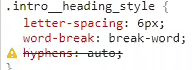
В зависимости от заданного значения он будет занимать определенный процент от родителя.

Хочется попробовать второй блок сделать так



свойство word-break отвечает за перенос слов.

hyphens работает в firefox, перенос слов с тире-



Вот хороший способ прижать футор

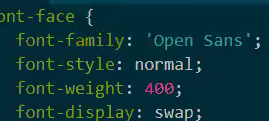


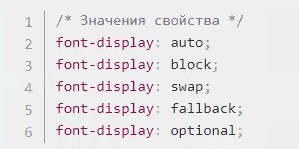
Взято отсюда <https://ishadeed.com/article/viewport-units/#2-second-solution-flexbox-and-viewport-units-recommended>

**Вебинар адаптивный CSS.**

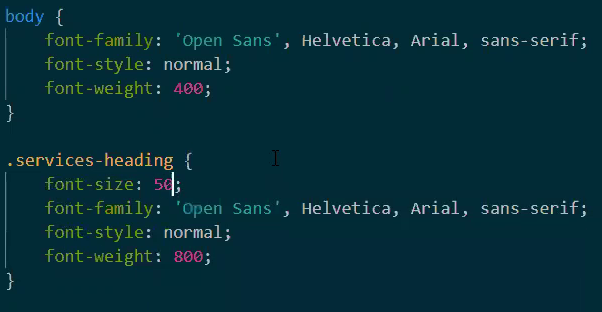
Когда скачиваешь шрифт локально, то скачиваются все начертания.

Свойство font-display, отвечает за переключение между шрифтами если основной еще не загрузился.





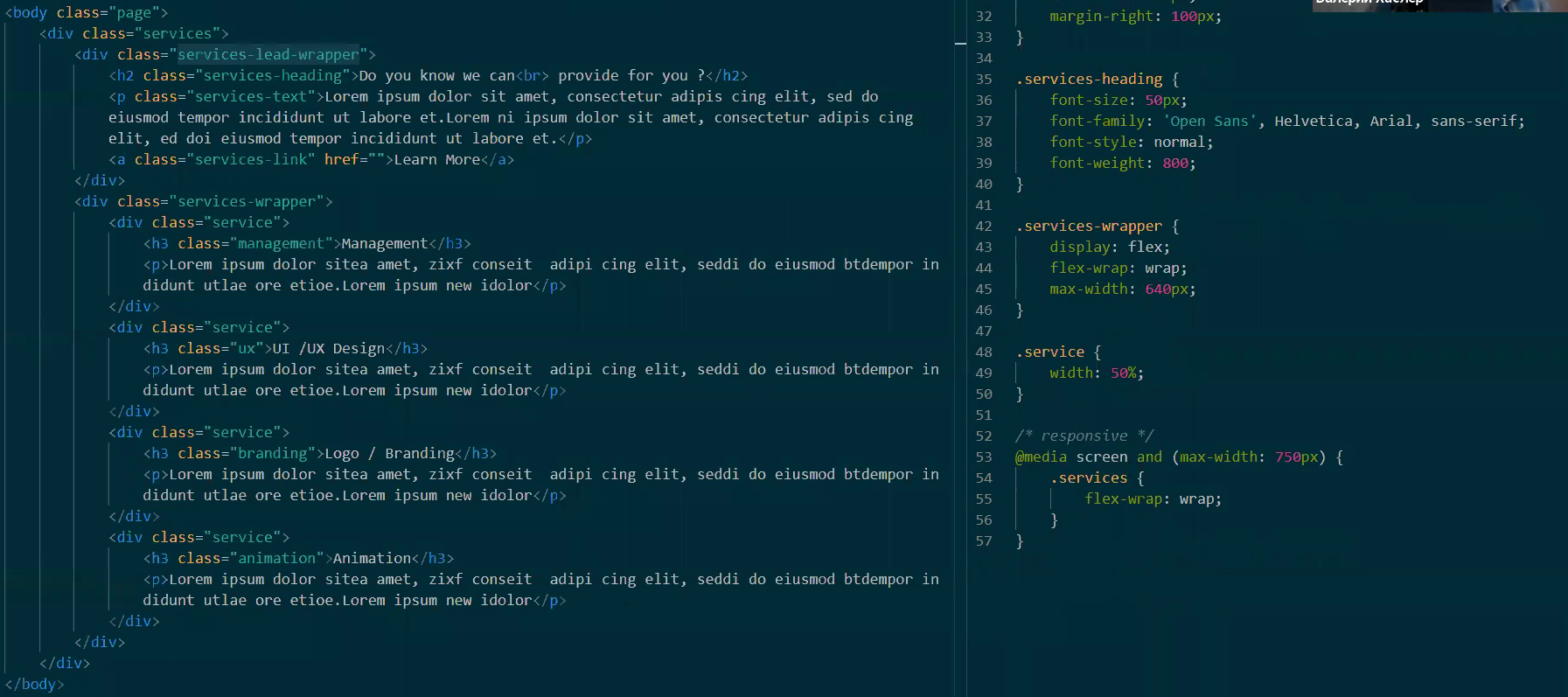
Также мы можем сделать подключенный шрифт жирнее, изменяя его вес.



К примеру у заголовков по умолчанию 600 – bold, а не 800 – extrabold.

margin auto будет работать только там, где задана максимальная ширина. Если ширина по умолчанию 100% как у div например, то не может быть никакого выравнивания. Только если есть пустое место.

Надо еще немного поиграться с этим кодом.­­



Медиа запросы писать только внизу. Иначе могут быть проблемы.

Flex-wrap лучше вообще не задавать изначально, а уже на определенной ширине включать через запросы.