JAVARUSH

Путь к карьере Frontend Fullstack разработчика

Модуль 1. WEB CORE

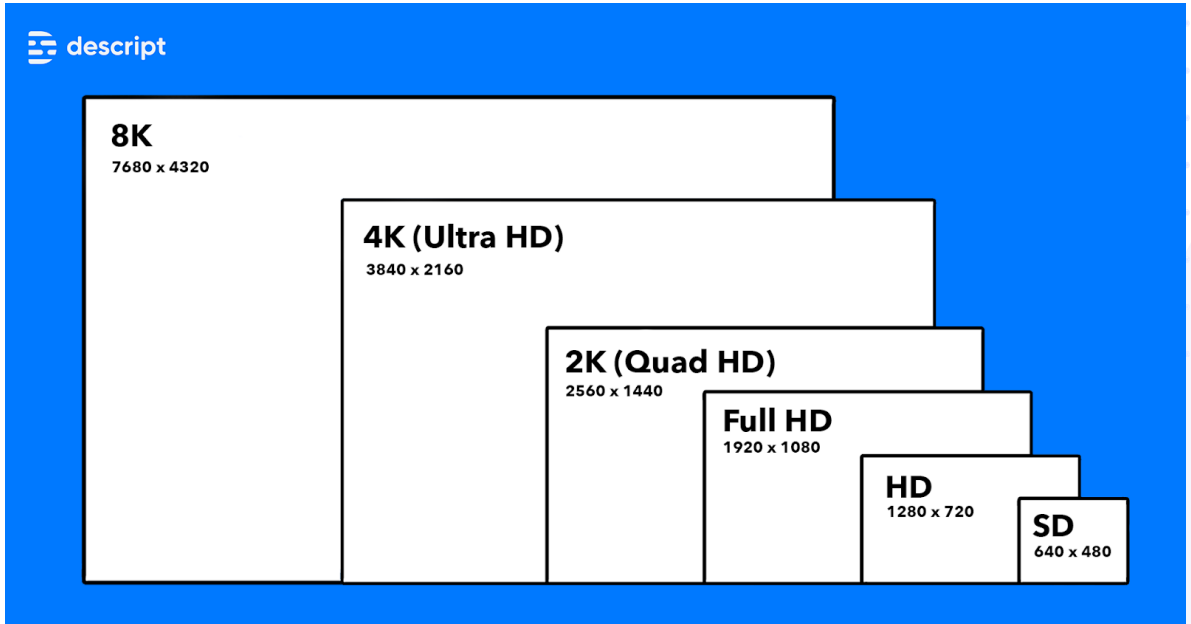
Уровень 13. Отзывчивая верстка.
Отзывчивая графика. Часть 2.

Адаптивные изображения для Retina-дисплеев

Современные устройства с экранами высокой плотности пикселей, такие как Retina-дисплеи, требуют особого подхода к оптимизации изображений. На таких экранах обычные изображения могут выглядеть размытыми или пикселизированными.

Проблема:

- На экранах с высокой плотностью пикселей один пиксель изображения растягивается на несколько физических пикселей экрана.



JAVARUSH

Атрибуты srcset и sizes

- Используйте атрибуты srcset и sizes для предоставления браузеру нескольких версий изображения с разным разрешением.
- Браузер автоматически выберет наиболее подходящее изображение, учитывая плотность пикселей устройства и размеры экрана.

```
<img
src="images/default.jpg"
srcset="images/image-1x.jpg 1x, images/image-2x.jpg 2x, images/image-3x.jpg 3x"
sizes="(max-width: 600px) 100vw, (max-width: 1200px) 50vw, 33vw"
alt="Пример изображения для Retina-дисплеев">
```

Объяснение:

- srcset: Указываем три версии изображения:
  - image-1x.jpg для обычных экранов (1x).
  - image-2x.jpg для экранов с удвоенной плотностью пикселей (2x).
  - image-3x.jpg для экранов с утроенной плотностью пикселей (3x).
- sizes: Указываем, какую ширину должно занимать изображение в зависимости от ширины экрана:
  - 100% ширины экрана, если ширина экрана меньше или равна 600px.
  - 50% ширины экрана, если ширина экрана от 601px до 1200px.
  - 33% ширины экрана, если ширина экрана больше 1200px.

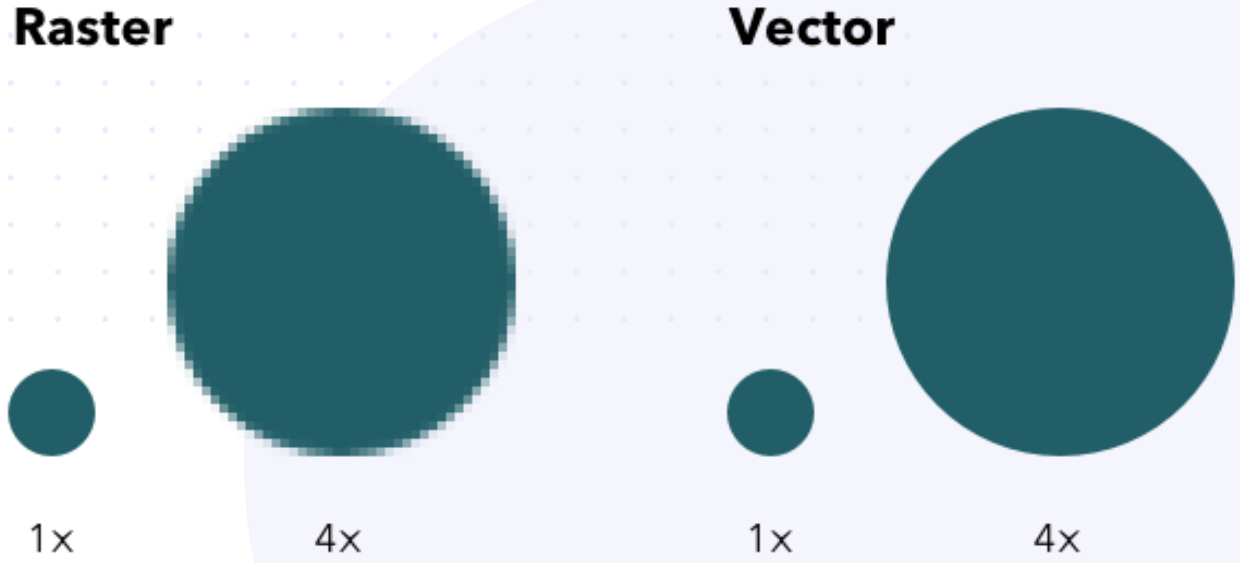
Векторная графика (SVG)

SVG расшифровывается как scalable vector graphics, «масштабируемая векторная графика». Это значит, перед нами векторное изображение, которое можно масштабировать без потери качества. Формат основанный на математических описаниях фигур и линий (технологии XML).

Это делает их идеальным выбором для экранов высокой плотности пикселей (Retina-дисплеи), где обычные изображения могут выглядеть размытыми или пикселизированными.

Чаще всего svg-иконки или изображения вам будет предоставлять дизайнер или вы сможете найти их на различных стоках:

- flaticon.com
- flowbite.com
- icomoon.io



JAVARUSH

Как подключать SVG-изображения

SVG-изображения можно подключить к веб-странице несколькими способами, каждый из которых имеет свои плюсы и минусы.

1. Тег <img>:

- **Простота:** Самый простой и распространённый способ.
- **Управление размерами:** Легко изменять размеры изображения с помощью CSS.
- **Ограничения:** Нельзя стилизовать внутренние элементы SVG через CSS.
- <img class="icon" src="img/svg/logo.svg" alt="logo">

2. background-image:

- **Для неконтентных изображений:** Подходит для фонов, иконок и декоративных элементов.
- **Ограничения:** Аналогично тегу <img>, нельзя стилизовать SVG через CSS.
- x

3. Вставка в HTML:

- **Полный контроль:** Позволяет стилизовать и анимировать SVG с помощью CSS.
- **Поддержка анимации:** Можно создавать динамические эффекты.
- **Неудобство редактирования:** При изменении SVG нужно править код во всех местах его вставки.

Ссылки и примеры для глубокого погружения:

- https://github.com/yoksel
- https://codepen.io/yoksel/pens/public
- https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial

JAVARUSH

Адаптивные фоновые изображения с помощью CSS

Для фоновых изображений можно использовать **медиа-запросы и свойства CSS**, чтобы загружать разные версии изображений в зависимости от плотности пикселей устройства.
Браузер загрузит только то изображение, которое подходит для конкретного устройства, экономя трафик и ускоряя загрузку страницы.

```
.background {
   background-image: url('images/background-1x.jpg'); /* Базовое изображение */
}

@media only screen and (min-resolution: 2dppx) {
    .background {
        background-image: url('images/background-2x.jpg'); /* Изображение для экранов с
плотностью 2x */
    }
}

@media only screen and (min-resolution: 3dppx) {
    .background {
        background-image: url('images/background-3x.jpg'); /* Изображение для экранов с
плотностью 3x */
    }
}
```

JAVARUSH

Элемент <picture>

Он позволяет указать несколько источников изображения с различными условиями, и браузер сам выберет наиболее подходящий вариант.

```
<picture>
<source srcset="images/image-3x.jpg" media="(min-resolution: 3dppx)">
<source srcset="images/image-2x.jpg" media="(min-resolution: 2dppx)">
<img src="images/image-1x.jpg" alt="Пример изображения для Retina-дисплеев">
</picture>
```

Как это работает:

- Браузер проверяет каждое условие в тегах <source> сверху вниз.
- Если условие выполняется,, браузер загружает изображение, указанное в атрибуте srcset этого тега.
- Если ни одно условие не выполняется, или браузер не поддерживает <picture>, загружается изображение, указанное в теге <img>.

JAVARUSH

Адаптация текста к любому экрану

Отзывчивые шрифты — это шрифты, которые автоматически подстраиваются под размеры экрана устройства, на котором просматривается веб-страница. Это достигается с помощью специальных относительных единиц измерения, таких как vw, vh, vmin и vmax.

Основные единицы:

- vw **(viewport width):** 1 vw равен 1% от ширины области просмотра браузера.
- vh **(viewport height):** 1 vh равен 1% от высоты области просмотра браузера.
- vmin **(viewport minimum):** 1 vmin равен 1% от наименьшего значения между шириной и высотой области просмотра.
- vmax **(viewport maximum):** 1 vmax равен 1% от наибольшего значения между шириной и высотой области просмотра.

```
h1 {
    font-size: 4vw; /* Заголовок будет занимать 4% ширины вьюпорта*/
}

p {
    font-size: 2vmin; /* Размер абзаца будет 2% от наименьшей стороны вьюпорта*/
}
```

JAVARUSH

Адаптивные таблицы

Таблицы — это отличный способ представить структурированные данные, но на маленьких экранах они могут стать настоящей головной болью для пользователей. К счастью, CSS позволяет нам создавать адаптивные таблицы, которые будут корректно отображаться на любых устройствах.

Пример: https://codepen.io/AllThingsSmitty/pen/MyqmdM

Прокрутка (Scroll)

Один из простых и эффективных способов сделать таблицу адаптивной — это добавить горизонтальную прокрутку. Таким образом, даже если таблица слишком широкая для экрана, пользователь сможет прокрутить её содержимое и увидеть все данные.

Как это сделать:

1. **Оберните таблицу в контейнер:** Создайте или другой блочный элемент, который будет служить контейнером для таблицы.
2. **Задайте ширину контейнера:** Установите ширину контейнера на 100%, чтобы он занимал всю доступную ширину экрана.
3. **Добавьте горизонтальную прокрутку:** Примените свойство overflow-x: auto; к контейнеру. Это позволит содержимому таблицы прокручиваться по горизонтали, если оно не помещается в контейнер.

```
.table-container {
    overflow-x: auto; /* Добавляем горизонтальную прокрутку */
    width: 100%;
}

table {
    width: 100%; /* Таблица занимает всю ширину контейнера */
    border-collapse: collapse; /* Убираем двойные границы между ячейками */
}

th, td {
    border: 1px solid #ccc;
    padding: 8px;
    text-align: left;
}
```

Перенос в блоки (Stacking)

Перенос в блоки — это еще один способ сделать таблицы удобными для просмотра на маленьких экранах. Вместо того чтобы сжимать или обрезать колонки, мы превращаем каждую строку таблицы в отдельный блок, который отображается вертикально.

Как это работает:

1. **Медиа-запросы:** Используем медиа-запрос, чтобы применить стили только на маленьких экранах (например, @media (max-width: 600px)).
2. **Блочное отображение:** Устанавливаем display: block для всех элементов таблицы (table, thead, tbody, th, td, tr), чтобы они стали блочными элементами и располагались друг под другом.
3. **Скрываем заголовки:** Скрываем заголовки столбцов (th), так как они будут дублироваться перед каждой ячейкой.
4. **Добавляем псевдоэлементы:** Используем псевдоэлементы ::before для ячеек данных (td), чтобы добавить перед каждым значением название соответствующей колонки. Это помогает сохранить контекст данных и делает таблицу более понятной.
5. **Стилизуем ячейки:** Применяем стили к ячейкам данных (td), чтобы они выглядели как отдельные блоки с заголовками и значениями.

JAVARUSH

Трансформация в карточки (Card Layout)

Этот метод превращает каждую строку таблицы в отдельную карточку на маленьких экранах. Такой подход делает данные более удобными для чтения и восприятия на мобильных устройствах.

Как это работает:

1. **Медиа-запросы:** Применяем стили только на маленьких экранах (например, @media (max-width: 600px)).
2. **Блочное отображение:** Устанавливаем display: block для всех элементов таблицы, чтобы они стали блоками и располагались друг под другом.
3. **Скрываем заголовки:** Скрываем заголовки столбцов (th), так как они будут дублироваться в каждой карточке.
4. **Стилизуем строки:** Добавляем отступы и границы к строкам (tr), чтобы они выглядели как отдельные карточки.
5. **Добавляем псевдоэлементы:** Используем псевдоэлементы ::before для ячеек данных (td), чтобы добавить перед каждым значением название соответствующей колонки.
6. **Стилизуем ячейки:** Убираем границы у ячеек (td) и добавляем отступы между названием колонки и значением.

JAVARUSH

Использование CSS Frameworks

CSS-фреймворки, такие как Bootstrap, предлагают готовые решения для создания адаптивных таблиц, что значительно упрощает процесс разработки и обеспечивает кроссбраузерную совместимость.

Преимущества использования фреймворков:

- **Простота:** Не нужно писать собственные CSS-правила для адаптации таблиц.
- **Кроссбраузерность:** Фреймворки обеспечивают корректное отображение таблиц в разных браузерах.
- **Готовые стили:** Фреймворки предоставляют набор стилей для оформления таблиц, что экономит время разработки.

Недостатки:

- **Зависимость от фреймворка:** Вы привязаны к стилям и возможностям конкретного фреймворка.
- **Размер файлов:** Фреймворки могут увеличивать размер страницы из-за дополнительных CSS и JavaScript файлов.

Flexbox и отзывчивость

Flexbox позволяет создавать гибкие и адаптивные макеты. Он особенно удобен для расположения элементов в строку или столбец, а также для создания адаптивных колонок, которые подстраиваются под размер экрана.

Создание адаптивных колонок

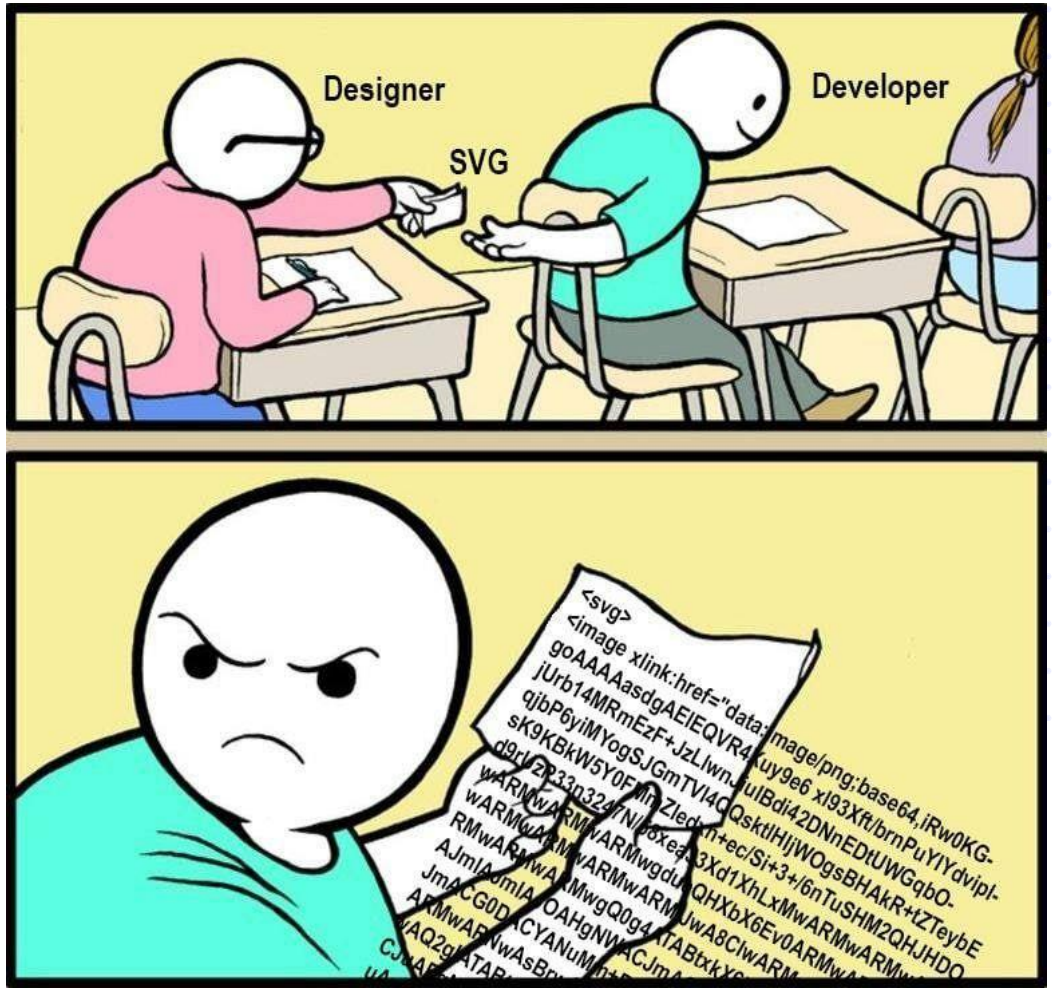
- flex-wrap: wrap: Позволяет элементам переноситься на новую строку, если они не помещаются в одну.
- flex-grow: Определяет, насколько элемент может расширяться, чтобы занять свободное пространство.
- flex-shrink: Определяет, насколько элемент может сжиматься, если места не хватает.
- flex-basis: Задает начальный размер элемента перед распределением свободного пространства.

```
.flex-container {
    display: flex;
    flex-wrap: wrap; /* Разрешаем перенос элементов */
    gap: 10px; /* Добавляем промежутки между элементами */
}

.flex-item {
    flex: 1 1 200px; /* Элементы могут расти и сжиматься, начальная ширина 200px */
    background-color: #3498db;
    color: white;
    padding: 20px;
    text-align: center;
}
```

JAVARUSH

Vectors are just great



JAVARUSH

Домашнее задание

Уровень 13. Отзывчивая верстка.
Отзывчивая графика.