Оптимизация frontend

Борис Ребров

ЦРИТО
Центр Развития
ИТ-Образования

МФТИ

@mail.ru
group

Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях

Коротко о курсе

- Три независимых модуля
- Семинары между модулями
- Больше времени на ДЗ
- Меньше штрафов
- Новый препод
- Новый чатик

Для чего нужно оптимизировать?

- Пользовательский опыт
  - снижение скорости загрузки страницы до 50% сокращает конверсию
  - понижение быстродействия страницы так же увеличивает число отказов
- SEO
  - скорость загрузки и быстродействие страниц влияет на ранжирование в поиске
- Нагрузка на сервера
  - количество запросов и объем передаваемых данных прямо влияют на производительность сервера

Что можно оптимизировать?

- Документ
  - Количество элементов
  - Сложность/специфичность селекторов
  - Периодичность пересчета/перерисовки
- Сетевое взаимодействие
  - Блокирующие ресурсы
  - Размер ресурсов
  - Количество запросов
- Вычисления
  - Сложные манипуляции с данными
  - Неоптимальное использование обработчиков событий

Документ

- O (Nodes * Selectors * Depth)
- Сокращаем количество элементов до минимально необходимого
- Не используем недостаточно специфичные комбинаторы

```
1. .my-class span {color: red}
2. .my-class > span {color: red}
```

```
1. <html>
2.     <div>
3.         <span>text</span>
4.     </div>
5.     <div class="my-class">
6.         <span>red text</span>
7.     </div>
8. </html>
```

Документ

- Избегайте повторного вычисления стилей и пересчета Layout
- Триггеры пересчета Layout (reflow)
  - Изменение «геометрических свойств» элемента
  - Изменение стилей или свойств, влияющий на стили (https://csstriggers.com/)
  - Обращение к свойствам элемента, измененным после последнего пересчета

```
1.  node.classList.add("my-class");
2.  console.log(node.offsetHeight);
```

  - И многое другое: http://bit.ly/reflows
- Используйте «слои», но не злоупотребляйте ими
  - will-change: *value;*
  - transform: translateZ(0);
- Ограничивайте размеры областей перерисовки и ее сложность

Документ

- BEM

```
<form class="search-form">
    <div class="search-form__content">
        <input class="search-form__input">
        <button class="search-form__button">Найти</button>
    </div>
</form>
```

- Css-modules

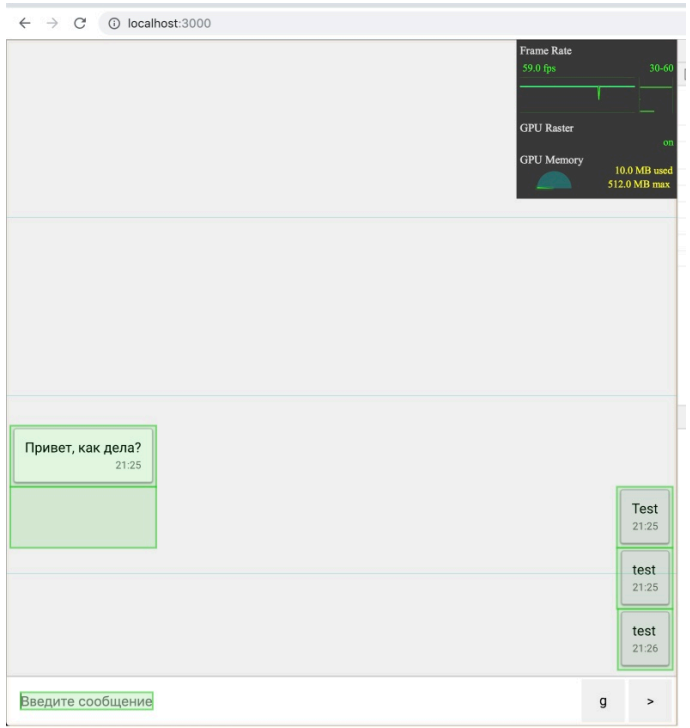
```
import styles from "./style.css";
// import { className } from "./style.css";

element.innerHTML = '<div class="' + styles.className + '">';
```

//для CRA достаточно добавить .module к имени стилевого файла

Документ

- Используйте инструменты анализа производительности
  http://bit.ly/render-settings
  http://bit.ly/perfomance-tools



Сетевое взаимодействие

- Загрузка скриптов и стилей может откладывать момент отрисовки
- Инлайн критических ресурсов
- Отложенная загрузка
  - defer, async, preload, prefetch
  - js загрузчики

Сетевое взаимодействие

- Загрузка скриптов и стилей может откладывать момент отрисовки
- Инлайн критических ресурсов
- Отложенная загрузка
  - defer, async, preload, prefetch
  - js загрузчики

Сетевое взаимодействие

- JS-бандлы

```
./src
├── main.js
└── component.js
```

```
./dist
└── bundle.js
```

Сетевое взаимодействие

- CSS-спрайты

```
1.  .foo {
2.      background: url('../assets/gift.png?sprite');
3.  }
```

```
1.  .foo {
2.      background: url(/sprite.png?[hash]) no-repeat;
3.      background-position: -100px -0px;
4.  }
```

Сетевое взаимодействие

- SVG-спрайты
  http://bit.ly/svg-sprite

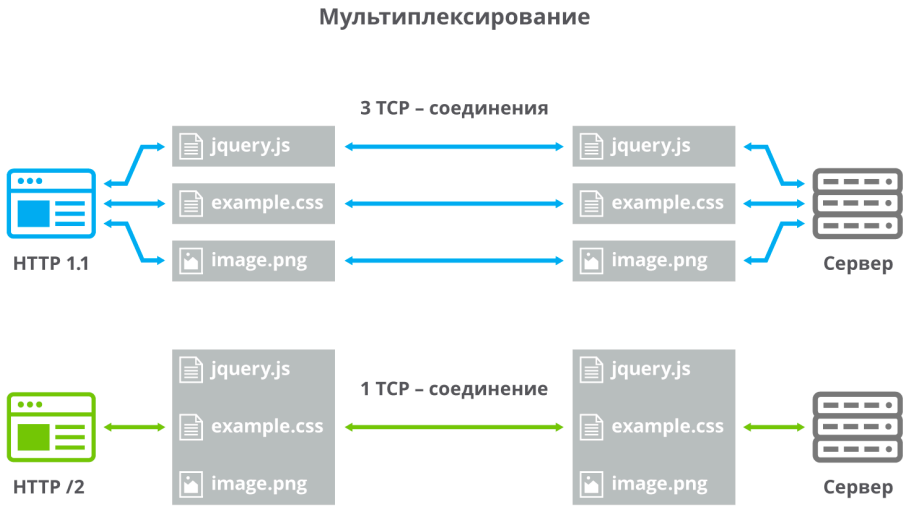
```
<svg version="1.1"
    xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
>
    <symbol id="first" viewBox="0 0 64 64">…</symbol>
    <symbol id="second" viewBox="0 0 64 64">…</symbol>
</svg>
```

```
<svg>
  <use xlink:href="#first"></use>
</svg>

<svg>
  <use xlink:href="#second"></use>
</svg>
```

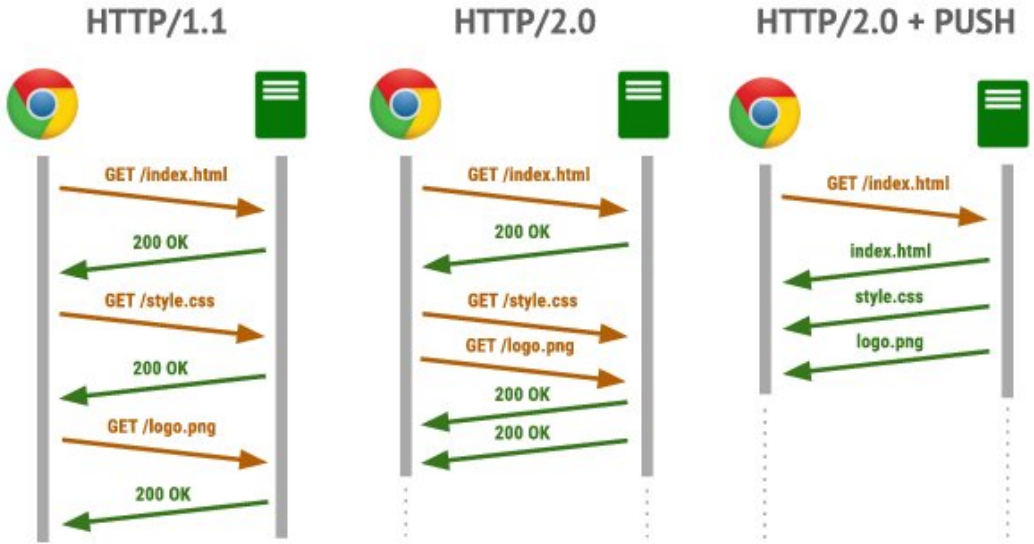
Сетевое взаимодействие

- HTTP/2
  - одно соединение для нескольких запросов
  - в отличие от Keep-alive ответы могут приходить одновременно



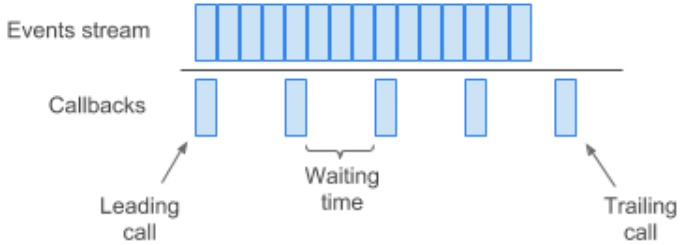
Сетевое взаимодействие

- HTTP/2 Sever Push
  - позволяет превентивно отправлять клиенту необходимые ресурсы
  - немного ломает логику кеширования на клиенте



Вычисления

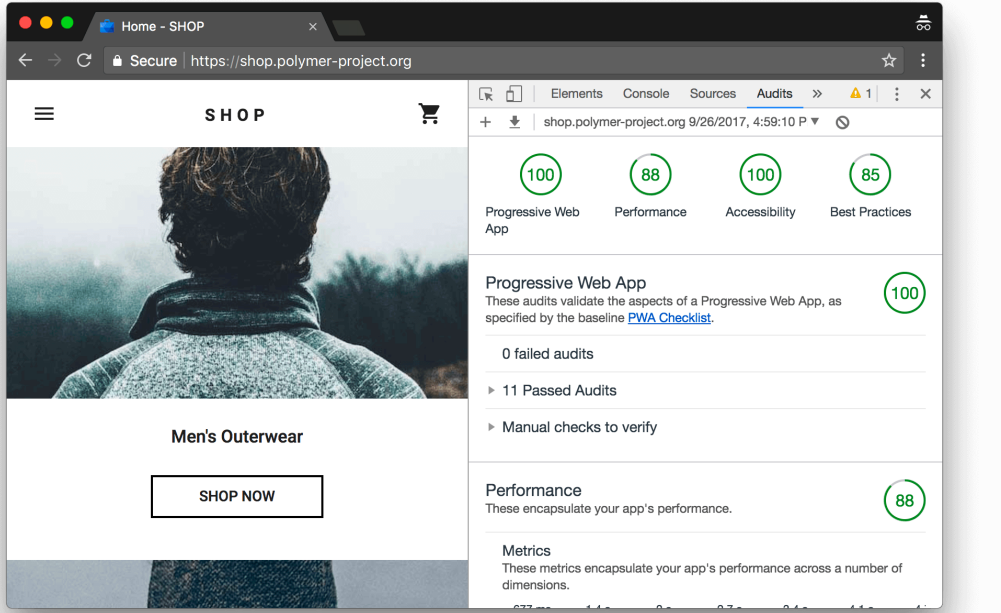
- throttling – вызов функции не чаще чем один раз за определенный интервал времени



- debouncing – вызов функции один раз, для серии вызовов происходящих чаще, чем заданный интервал времени

Сетевое взаимодействие

- Lighthouse
  http://bit.ly/googl-lighthouse



Домашнее задание № 1

- Реализовать в вашем приложении виртуальную клавиатуру-emoji, иконки объединить в спрайт

Срок сдачи

- до 7 марта

Спасибо за внимание!