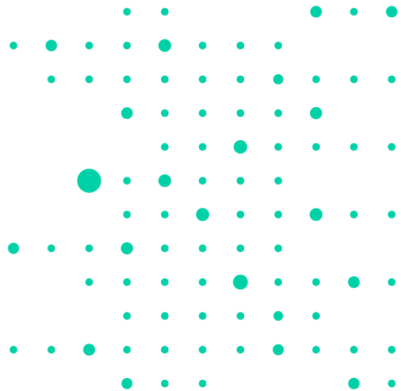




## Front-end overview.

Working with timers, `setTimeout` and `setInterval`, `localStorage`, `sessionStorage`



# Планирование: `setTimeout` и `setInterval`

---

Для этого существуют два метода:

**`setTimeout`** позволяет вызвать функцию **один раз** через определённый интервал времени.

**`setInterval`** позволяет вызывать функцию **регулярно**, повторяя вызов через определённый интервал времени.

# Планирование: setTimeout и setInterval

---

## setTimeout

Синтаксис:

```
let timerId = setTimeout(func|code, [delay], [arg1], [arg2], ...)
```

# Планирование: `setTimeout` и `setInterval`

---

Параметры:

`func|code`

Функция или строка кода для выполнения. Обычно это функция. По историческим причинам можно передать и строку кода, но это не рекомендуется.

`delay`

Задержка перед запуском в миллисекундах (1000 мс = 1 с). Значение по умолчанию – 0.

`arg1, arg2 ...`

Аргументы, передаваемые в функцию (не поддерживается в IE9-)

# Планирование: setTimeout и setInterval

---

Метод setInterval имеет такой же синтаксис как setTimeout:

Синтаксис:

```
let timerId = setInterval(func|code, [delay], [arg1], [arg2], ...)
```

# Планирование: `setTimeout` и `setInterval`

---

- Методы `setInterval(func, delay, ...args)` и `setTimeout(func, delay, ...args)` позволяют выполнять `func` регулярно или только один раз после задержки `delay`, заданной в мс.
- Для отмены выполнения необходимо вызвать `clearInterval/clearTimeout` со значением, которое возвращают методы `setInterval/setTimeout`.
- Вложенный вызов `setTimeout` является более гибкой альтернативой `setInterval`. Также он позволяет более точно задать интервал между выполнениями.

# Планирование: `setTimeout` и `setInterval`

---

- Планирование с нулевой задержкой `setTimeout(func,0)` или, что то же самое, `setTimeout(func)` используется для вызовов, которые должны быть исполнены как можно скорее, после завершения исполнения текущего кода.
- Браузер ограничивает 4-мя мс минимальную задержку между пятью и более вложенными вызовами `setTimeout`, а также для `setInterval`, начиная с 5-го вызова.
- Обратим внимание, что все методы планирования не гарантируют точную задержку.

# Планирование: `setTimeout` и `setInterval`

---

Например, таймер в браузере может замедляться по многим причинам:

- Перегружен процессор.
- Вкладка браузера в фоновом режиме.
- Работа ноутбука от аккумулятора.

Всё это может увеличивать минимальный интервал срабатывания таймера (и минимальную задержку) до 300 или даже 1000 мс в зависимости от браузера и настроек производительности ОС.



# localStorage, sessionStorage

---

- **sessionStorage** - будет очищен как только пользователь покинет сайт.
- **localStorage** - после того, как пользователь покинет сайт, продолжит свое существование.

# localStorage, sessionStorage

---

Основные особенности **localStorage**:

- Этот объект один на все вкладки и окна в рамках источника (один и тот же домен/протокол/порт).
- Данные не имеют срока давности, по которому истекают и удаляются. Сохраняются после перезапуска браузера и даже ОС.

# localStorage, sessionStorage

---

Объект **sessionStorage** используется гораздо реже, чем localStorage.

Свойства и методы такие же, но есть существенные ограничения:

- **sessionStorage** существует только в рамках текущей вкладки браузера.
  - Другая вкладка с той же страницей будет иметь другое хранилище.
  - Но оно разделяется между ифреймами на той же вкладке (при условии, что они из одного и того же источника).
- Данные продолжают существовать после перезагрузки страницы, но не после закрытия/открытия вкладки.

# localStorage, sessionStorage

---

Объекты хранилища localStorage и sessionStorage предоставляют одинаковые методы и свойства:

**setItem(key, value)** – сохранить пару ключ/значение.

**getItem(key)** – получить данные по ключу key.

**removeItem(key)** – удалить данные с ключом key.

**clear()** – удалить всё.

**key(index)** – получить ключ на заданной позиции.

**length** – количество элементов в хранилище.