Задание

> Написать html страницу с текстом "Сюрприз!". С помощью JavaScript выполнить 5 000 итераций, поменять фон на желтый и вывести текст "Сейчас будет еще один цветной сюрприз...", выполнить 50 000 итераций и поменять фон

Решение

В папке с ресурсами есть две папки **first version/** и **improved version/**

first version - начальный вариант решения improved version - улучшенный вариант со стилями

Пройдемся по каждому.

first-version/index.html

```
<html>
    <head><title>Background Color Changing Test</title></head>
    <body>
        <h1>СЮРПРИЗ!</h1>
        <script>
            for(var i = 0; i < 5000; i++){}
            document.bgColor = "yellow"
        </script>
        <div>Сейчас будет еще один цветной сюрприз...</div>
        <script>
            for(var i = 0; i < 50000; i++){}
            document.bgColor = "red"
        </script>
    </body>
</html>
```

▶ Анализ кода

> 1. Определяем документ через тег html
> 2. Внутри тега head объявляем title страницы
> 3. body - тело документа, тут пишется основная разметка страницы
> 4. пишем текст "Сюрпиз!" как заголовок h1 (для стиля)
> 5. JS скрипт пишется внутри тега script
> 6. Нам нужно выполнить 5000 итераций `for(var i = 0; i < 5000; i++) {}` Выполняется 5000 раз и меняется цвет фона на желтый `document.bgColor = "yellow"`
> 7. Выводим следующий текст
> 8. Еще один цикл выполняется 50000 раз и меняет цвет фона на красный `document.bgColor = "red"`

Данный код работает, но неправильно. Если открыть файл в браузере мы увидим конечный результат, а все выполненные действия нет. Дело в том что браузер читает каждый тег и создает их объект, читает скрипт и выполняет, после этого прорисовывает все это на экране. По этому видим только конечный результат, а не процесс выполнения. Если прорисовка происходила моментально мы получили то чего ожидали, но производительность браузера резко уменьшилась

Но есть решение!

В папке **improved-version/**

Код разделил на три файла:

```
index.html - HTML разметка страницы
css/
   style.css - CSS стили
js/
   script.js - JavaScript код
```

> Разделения на файлы хорош тем что код станет более понятым и лаконичным. Не придется для изменения например, javascript'а искать в куче тегов тег script, а просто открыть js/script.js

Теперь немного углубимся в код. В первую очередь заглянем index.html и разберем все по подробней

```
<html>
    <head>
        <title>Background Color Change Test</title>
        <link rel="stylesheet" href="css/style.css">
        <script src="js/script.js"></script>
    </head>
    <body>
        <div id="wrapper">
            <div id="surpise">Сюрприз!</div>
            <div id="next-surprise"></div>
            <div id="dynamic-container">
                <input class = "click-me" onclick = "onClickAction()" value="Нажми
на меня!" type="button">
                </div>
            </div>
        </body>
    </html>
```

только теги

Подключаем стили

```
<link rel="stylesheet" href="css/style.css">
```

Подкючаем скрипт

```
<script src="js/script.js"></script>
```

id для определения стилей (см. css/styles.css: #surprise)

```
<div id="surpise">Сюрприз!</div>
```

Здесь будет размещатся текст *"Сейчас будет еще один цветной сюрприз..."*. Оставил пустым, так как он выводится после. Поменяем его в JS-коде

```
<div id="next-surprise"></div>
```

dynamic-container

```
<div id="dynamic-container">
    <input class = "click-me" onclick = "onClickAction()" value="Нажми на меня!"
type="button">
</div>
```

тут немного сложновато, но понять в полне реально. Если запустить index.html в браузере, увидим кнопку "Нажми на меня", описанный тут

```
<input class = "click-me" onclick = "onClickAction()" value="Нажми на меня!"
type="button">
```

Так, если нажать на него, кнопка пропадает и в место него появляется счетчик. **id="dynamic-container"** нужен для того чтобы после нажатия на кнопку заменить его содержимое на счетчик, т.е.:

```
-dynamic-container
    -button
```

меняется на:

```
-dynamic-container
    -counter
```

Теперь сама кнопка

```
<input class = "click-me" onclick = "onClickAction()" value="Нажми на меня!"
type="button">
```

class="click-me" это класс описания стилей (см. css/style.css: .click-me)

onclick="onClickAction()" Данный атрибут говорит какое действие следует выполнить при нажатии на кнопку. В нашем случае это функция onClickAction() в файле js/script.js

наконец-то Javascript

Открываем файл js/script.js

```
function onClickAction() {

    container = elementById("dynamic-container")
    container.innerHTML = "<span id = \"counter\">0</span>"

    runLoop(changeCounter, function() {
        document.bgColor = "yellow"

        elementById("next-surprise").textContent = "Сейчас будет еще один цветной
сюрприз..."

        runLoop(changeCounter, function() {

            document.bgColor = "blue"

            elementById("surpise").textContent = "ВСЕ!"
            elementById("next-surprise").textContent = "Закончили, закрой вкладку
:)"

        }, 5000)
    }, 5000)
}

function runLoop(actionInIteration, actionAfterComplete, interations) {
        i = 0;
        var interval = setInterval(function(){
            if(i > interations) {
                clearInterval(interval)
                actionAfterComplete()
            }
            actionInIteration(i)
```

```
            i++
    }, 1)
}

function changeCounter(time) {
    elementById("counter").textContent = time
}

function elementById(id) {
    return document.getElementById(id)
}
```

Фунции changeCounter и elementById

Две функции **changeCounter** и **elementById** дополнительные фукнции.

changeCounter - Меняет значение счетчика на значение time

elementById - Возвращает элемент с по указанному id (чтобы не приходилось каждый раз писать `document.getElementById(id)`)

Фунция runLoop

Сначала давайте изучим функцию runLoop

```
function runLoop(actionInIteration, actionAfterComplete, interations) {
        i = 0;
        var interval = setInterval(function(){
            if(i > interations) {
                clearInterval(interval)
                actionAfterComplete()
            }
            actionInIteration(i)
            i++
    }, 1)
}
```

Функция принимает 3 аргумента

- actionInIteration
- actionActerComplete
- iterations

Фукнция выполняет цикл столько раз, сколько задано параметром **iterations**

При каждой итерации вызывает функцию **actionInIteration** с значением текущей итерации

После окончания цикла вызывает функцию **actionAfterComplete**

Задаем текущий индекс итерации

```
    i = 0;
```

Нельзя обновлять страницу через цилк for, поэтому используется страндартная фукнция JS анимации setInterval

setInterval(function, ms) принимает два аргумента, первый аргумент это функция, второй числовое значение. Выполняет переданную функция **function** каждую миллисекунду переданную в как аргумент **ms**. Возвращает свой экземпляр для манипуляций, сохранеям в переменную, например в `var interval`

```
var interval = setInterval(function(){

    if(i > interations) {
        clearInterval(interval)
        actionAfterComplete()
    }

    actionInIteration(i)
    i++
}, 1)
```

т.е. в нашем случае следующий код будет выполнятся каждую 1 миллисекунду.

```
if(i > interations) {
    clearInterval(interval)
    actionAfterComplete()
}
actionInIteration(i)
i++
```

Теперь разберем что происходит каждую 1мс

```
if(i > interations) {
    clearInterval(interval)
    actionAfterComplete()
}
```

Если количество итераций `i` превысит заданное колчество в аргументе `iterations` цикл прекращается, для этого остановки в функцию `clearInterval()` передается экземпляр возвращенный с `setInterval`

```
clearInterval(interval)
```

После завершения выполняется фукнция `actionAfterComplete`

Если `i` меньше чем `interations` вызывается функция `actionInIterations(i)` передав текущее значение `i`

Фунция onClickAction

Начинаем с самого начала. При нажатии на кнопку вызывается функция `onClickAction()`.

```
function onClickAction() {

    container = elementById("dynamic-container")
    container.innerHTML = "<span id = \"counter\">0</span>"

    runLoop(changeCounter, function() {
        document.bgColor = "yellow"

        elementById("next-surprise").textContent = "Сейчас будет еще один цветной
сюрприз..."

        runLoop(changeCounter, function() {

            document.bgColor = "blue"

            elementById("surpise").textContent = "ВСЕ!"
            elementById("next-surprise").textContent = "Закончили, закрой вкладку
:)"

        }, 5000)
    }, 5000)
}
```

Что тут происходит? Как говорил выше, при нажатии кнопка пропадает и в место него появляется счетчик.

Получаем объект dynamic-container

```
container = elementById("dynamic-container")
```

Меняем сожержимое контейнера с помощью свойства innerHTML

```
container.innerHTML = "<span id = \"counter\">0</span>"
```

Вызываем `runLoop` в качестве аргумента `actionInIteration` передаем функцию `changeCounter` чтобы при каждой итерации менял значение счетчика. А в качестве `actionAfterComplete` (запускается после завершения цикла) передаем анонимную функцию которая будет выполнятся каждую 1мс 500 раз

```
runLoop(changeCounter, function() {
    ...
}, 500)
```

содержание первой runLoop

После завершения первого цикла в 500 раз меняем фон на желтый

```
document.bgColor = "yellow"
```

получаем объект с id 'next-surprice' и меняем его текст с помощью свойства textContent

```
elementById("next-surprise").textContent = "Сейчас будет еще один цветной
сюрприз..."
```

апускаем еще один цикл в 5000 итераций. В качестве аргумента `actionInInteration` передаем функцию `changeCounter` чтобы менял счетчик

После завершения меняем опять меняем фон

```
runLoop(changeCounter, function() {
    // После
    document.bgColor = "blue"
}, 5000)
```

Концовка

Данную задачу можно было решить и по другому, более коротко и быстро использовав функцию setTimeout, но счетчика бы не было 😀

> Можете попробовать файл js/script2.js
>
> меняем

```
<script src="js/script.js"></script>
```

> на

```
<script src="js/script2.js"></script>
```

Думаю что было понятно, если не понятно всегда можете спросить. Удачи! 😀