Kotlin на платформе JavaScript

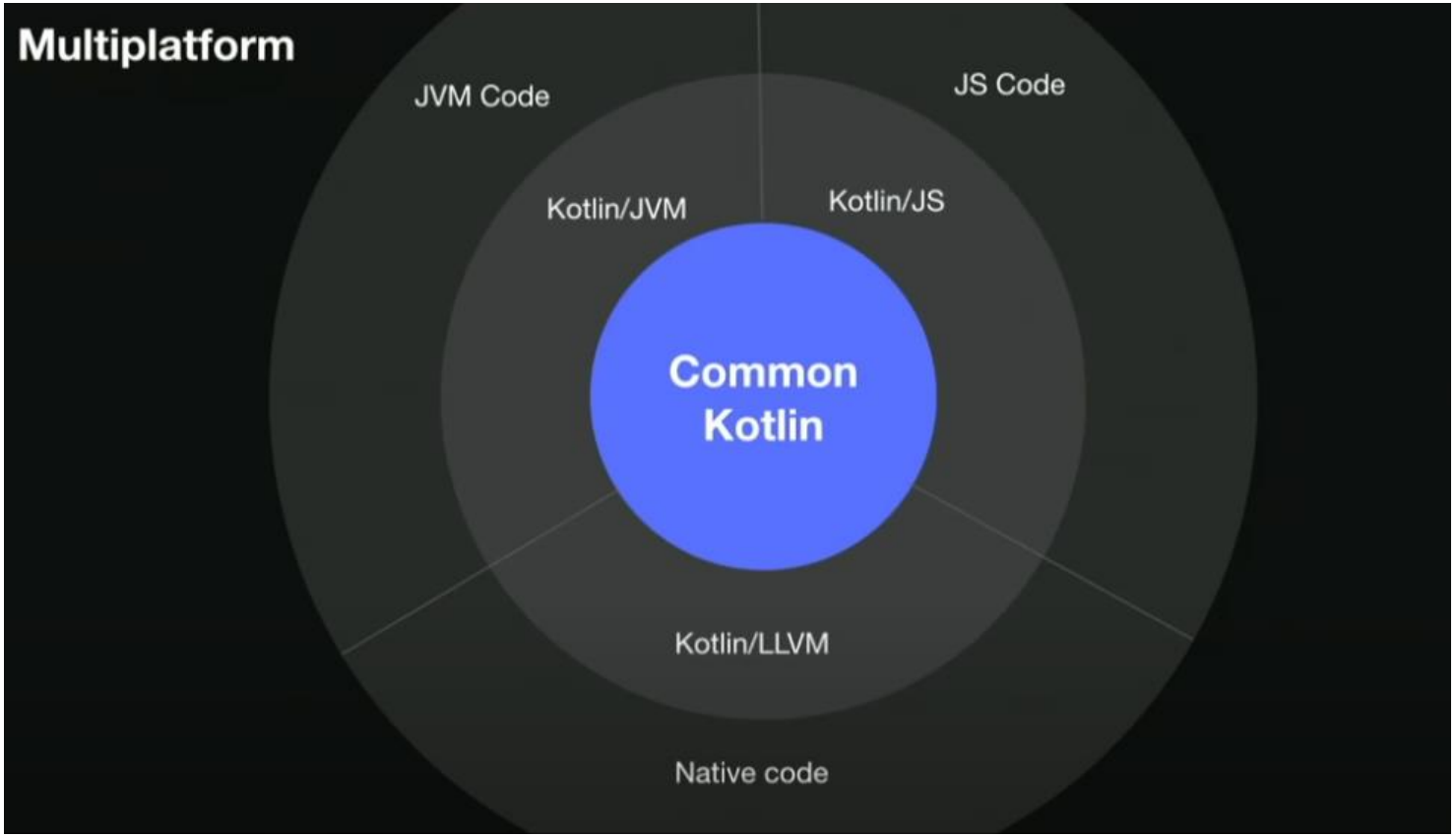
Романов Владимир Юрьевич

Декабрь 04, 2021

6. KOTLIN НА ПЛАТФОРМЕ JAVASCRIPT

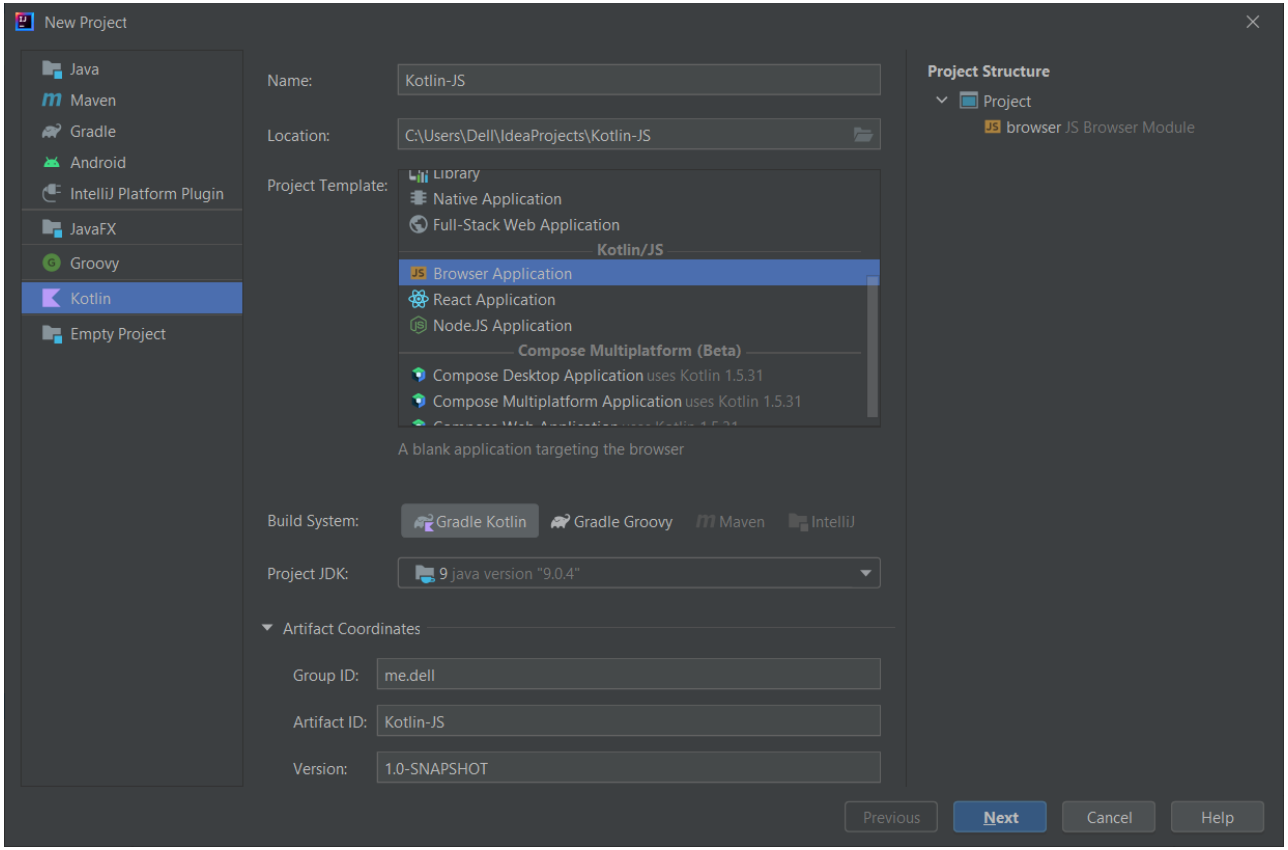
6.1 СОЗДАНИЕ ПРОЕКТА ДЛЯ ПЛАТФОРМЫ JAVASCRIPT

Платформа Kotlin/JS



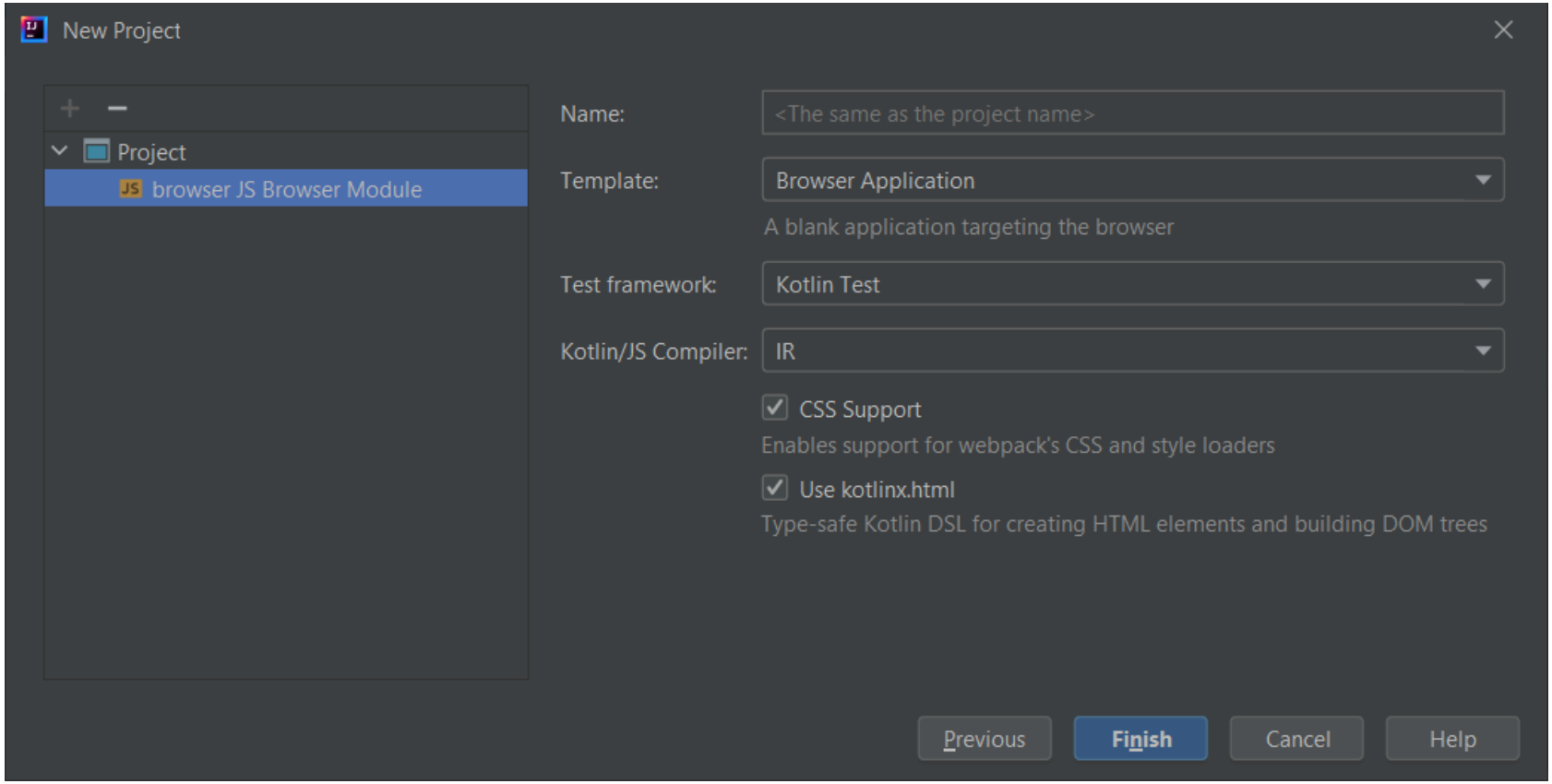
Создание проекта для платформы JavaScript

Kotlin/JS в браузере



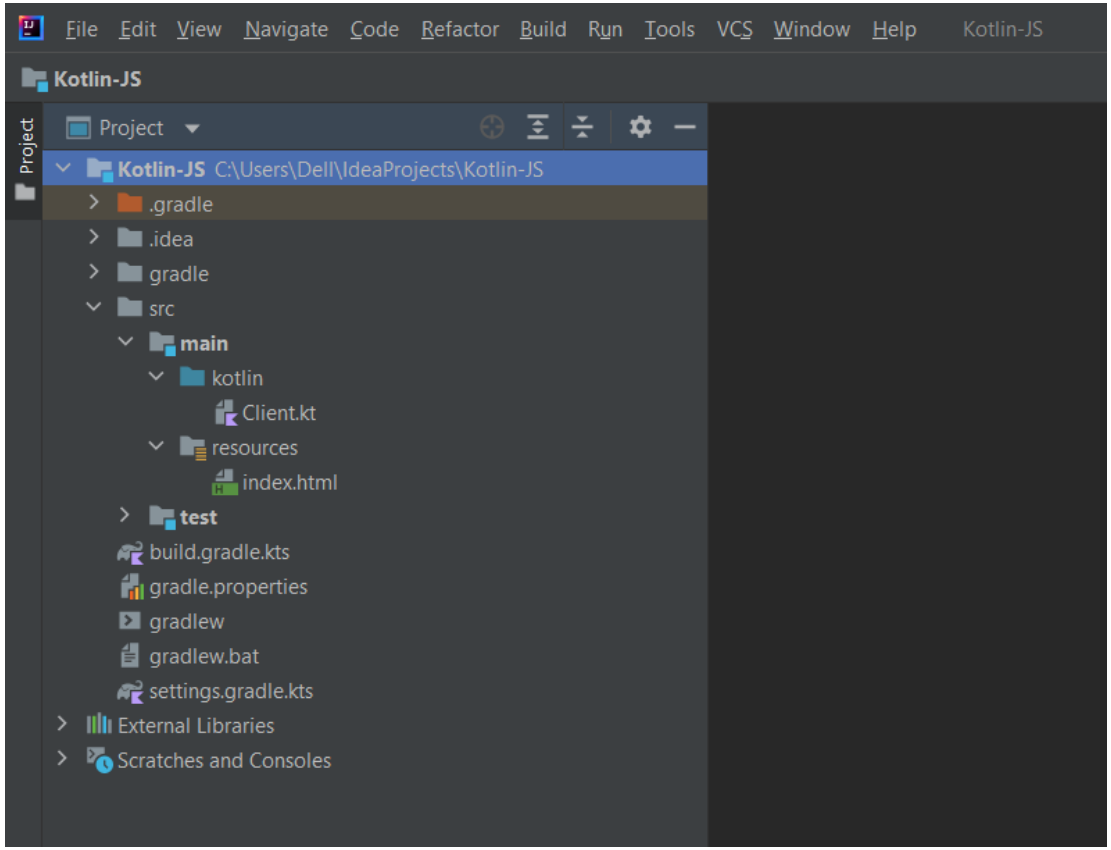
Kotlin/JS в браузере

Описание модуля Kotlin/JS в браузере



Описание модуля Kotlin/JS в браузере

Структура проекта



Структура проекта

Описание проекта *build.gradle.kts* (1)

```
plugins {
    kotlin("js") version "1.6.0"
}
group = "me.dell"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
    maven("https://maven.pkg.jetbrains.space/public/p/kotlinx-
html/maven")
}
dependencies {
    testImplementation(kotlin("test"))
    implementation("org.jetbrains.kotlinx:kotlinx-html:0.7.2")
}
```

Описание проекта *build.gradle.kts* (2)

```
kotlin {
    js(IR) {
        binaries.executable()
        browser {
            commonWebpackConfig {
                cssSupport.enabled = true
            }
        }
    }
}
```

Функция *kotlin* (использование)

```
plugins {
    kotlin("js") version "1.6.0"
}
```

Функция *kotlin* (объявление)

```
/**
 * Configures the
[kotlin][org.jetbrains.kotlin.gradle.dsl.KotlinJsProjectExtension]
extension.
 */
fun org.gradle.api.Project.`kotlin`(
    configure:
Action<org.jetbrains.kotlin.gradle.dsl.KotlinJsProjectExtension>
    ): Unit =
    (this as org.gradle.api.plugins.ExtensionAware)
        .extensions.configure("kotlin", configure)
```

Функция *version*

Использование:

```
plugins {
    kotlin("js") version "1.6.0"
}
```

Объявление:

```
/**
 * Specify the version of the plugin to depend on.
 *
 * Infix version of [PluginDependencySpec.version].
 */
infix fun PluginDependencySpec.version(version: String?)
    : PluginDependencySpec = version(version)
```

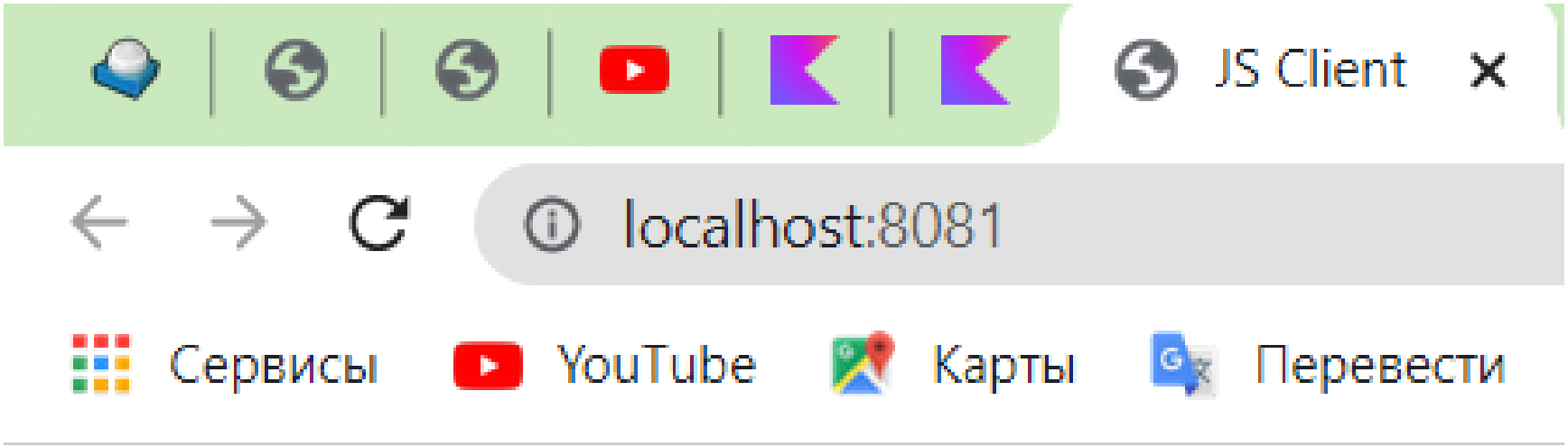
Ресурс *index.html*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>JS Client</title>
</head>

<body>
<script src="Kotlin-JS.js"></script>
<div id="root"></div>
</body>

</html>
```

Клиент *Client.kt*

```
import kotlinx.html.div
import kotlinx.html.dom.append
import org.w3c.dom.Node
import kotlinx.browser.document
import kotlinx.browser.window

fun main() {
    window.onload = { document.body?.sayHello() }
}
fun Node.sayHello() {
    append {
        div {
            +"Hello from JS"
        }
    }
}
```

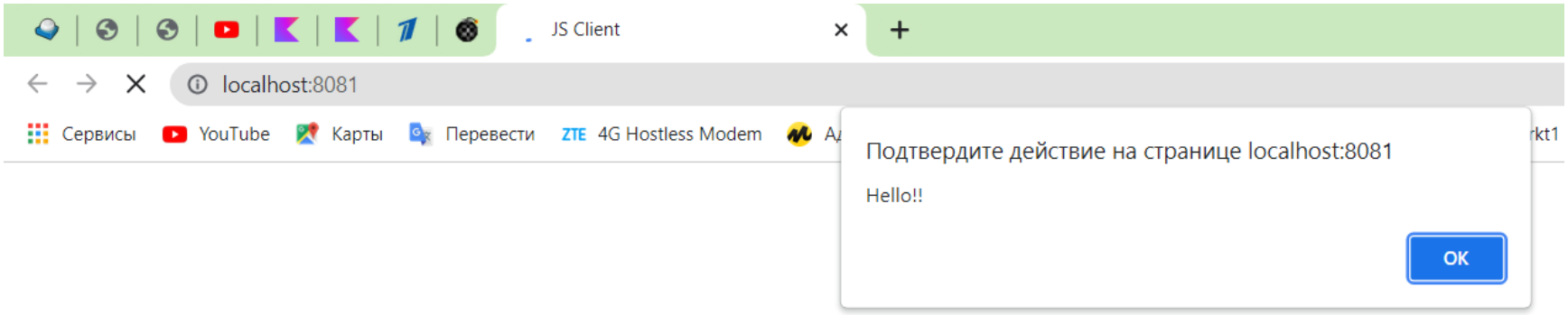
Клиент в браузере



Взаимодействие с Document Object Model (DOM)

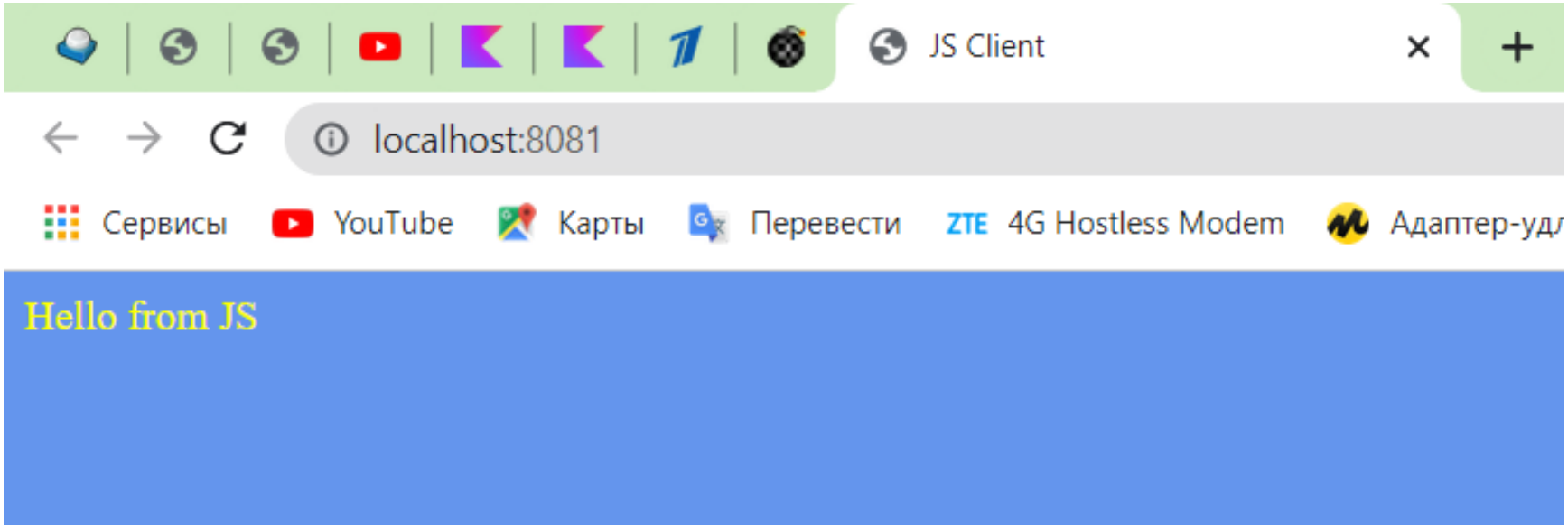
```
fun main() {
    document.bgColor = "6495ED"
    document.fgColor = "yellow"
    window.alert("Hello!!")

    window.onload = { document.body?.sayHello() }
}
```

Выдача сообщения



Выдача сообщения

Раскраска текста и фона страницы

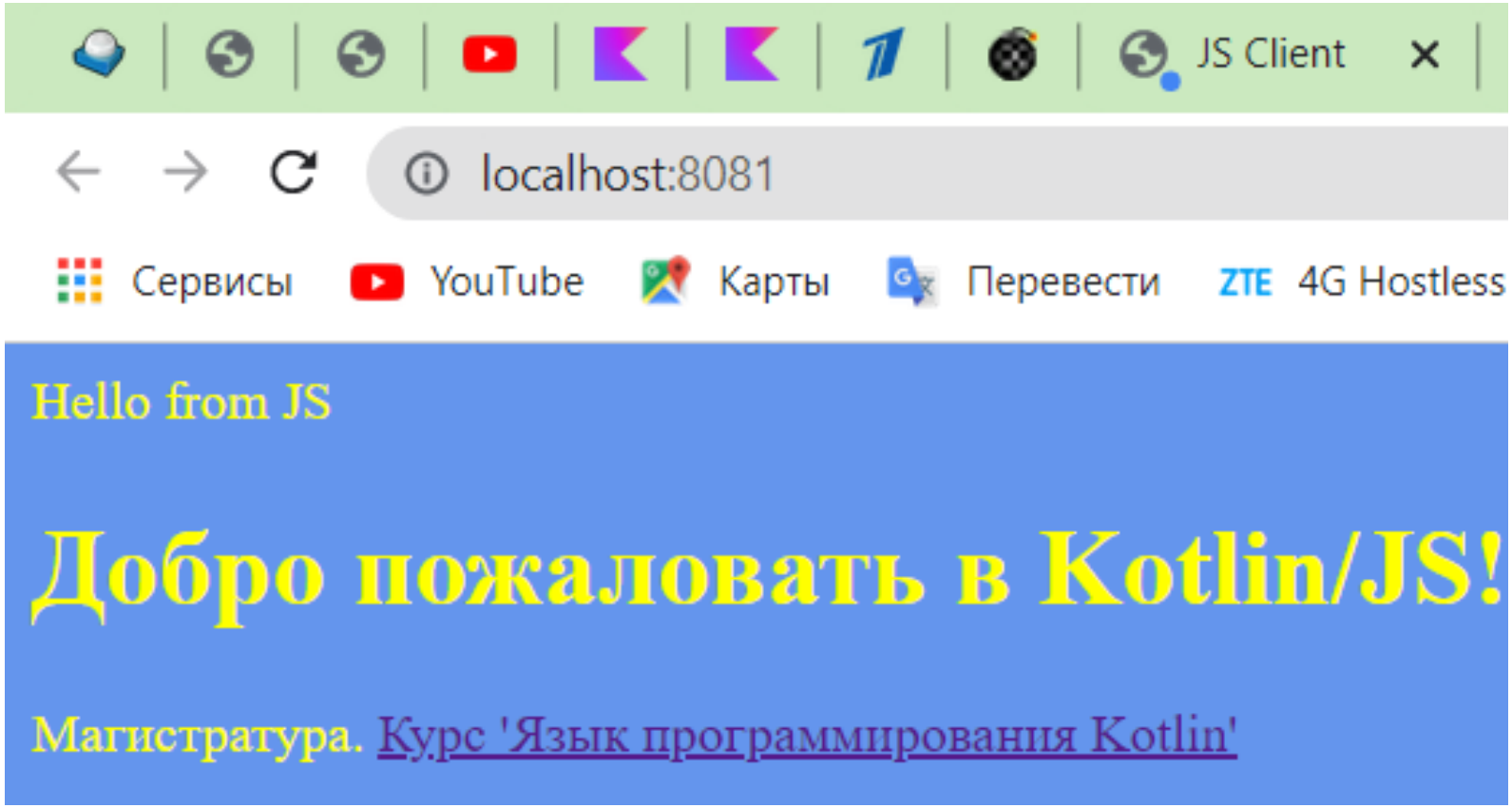


Раскраска текста

Использование типизированного *HTML*

```
window.onload = {
    document.body?.sayHello()
    document.body!!.append.div {
        h1 {
            +"Добро пожаловать в Kotlin/JS!"
        }
        p {
            +"Магистратура. "
            a("http://master.cmc.msu.ru/") {
                +"Курс 'Язык программирования Kotlin'"
            }
        }
    }
}
```

Использование типизированного *HTML* (в браузере)

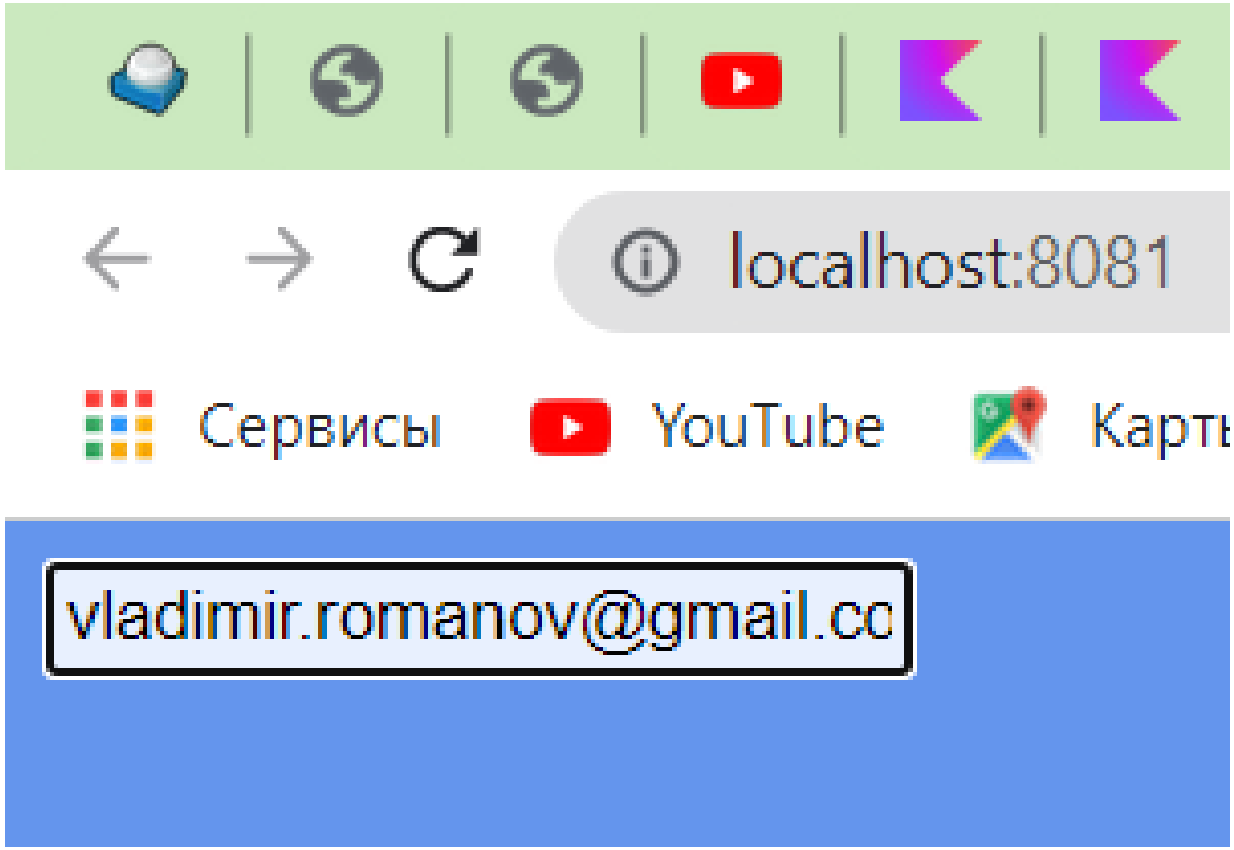


Использование типизированного *HTML*

Использование управляющих элементов

```
//    <input type="text" name="email" id="email"/>
    window.onload = {
        document.body!!.append.div {
            input {
                type = InputType.text
                name = "email"
                id = "email"
            }
        }
    }

    val email = document.getElementById("email") as
HTMLInputElement
    email.value = "vladimir.romanov@gmail.com"
```

Использование управляющих элементов (в браузере)



Использование управляющих элементов

Изменение содержимого элемента *HTML* (1)

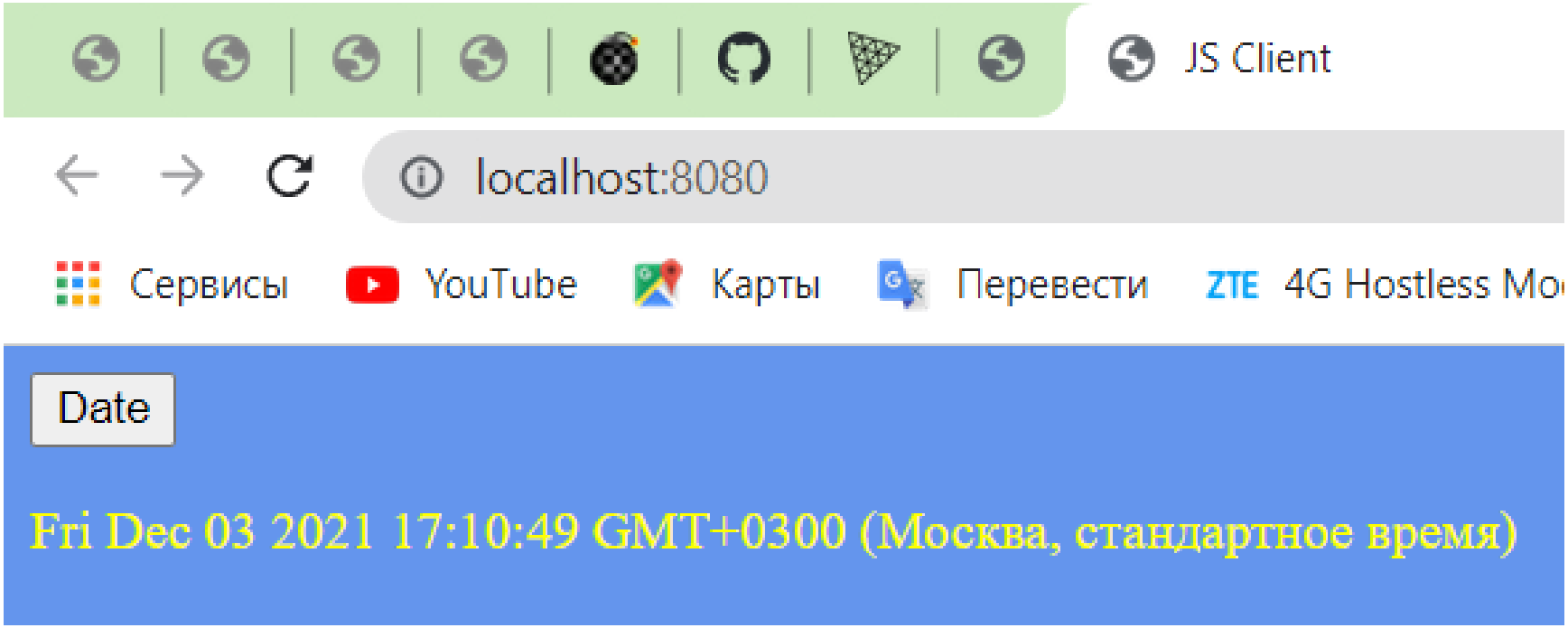
При нажатии на кнопку должна показываться текущая дата

```
fun main() {
    document.bgColor = "6495ED"
    document.fgColor = "yellow"

    window.onload = {
        document.body?.addDateButton()
    }
}
```

Изменение содержимого элемента *HTML* (2)

```
fun Node.addDateButton() {
    append {
        div {
            button {
                +"Date"
                type = ButtonType.button
                onClickFunction = { sayDate() }
            }
            p {  id = "demo" }
        }
    }
}
fun sayDate() {
    document.getElementById("demo")?.innerHTML
                                = Date().toString()
}
```

Изменение содержимого элемента *HTML* (в браузере)



Изменение содержимого элемента *HTML*. Текущая дата

Изменение атрибута элемента *HTML* (1)

При нажатии на кнопку должна включаться/выключаться лампа

```
fun main() {
    document.bgColor = "6495ED"
    document.fgColor = "yellow"

    window.onload = {
        document.body?.addToggleBulbButton()
    }
}
```

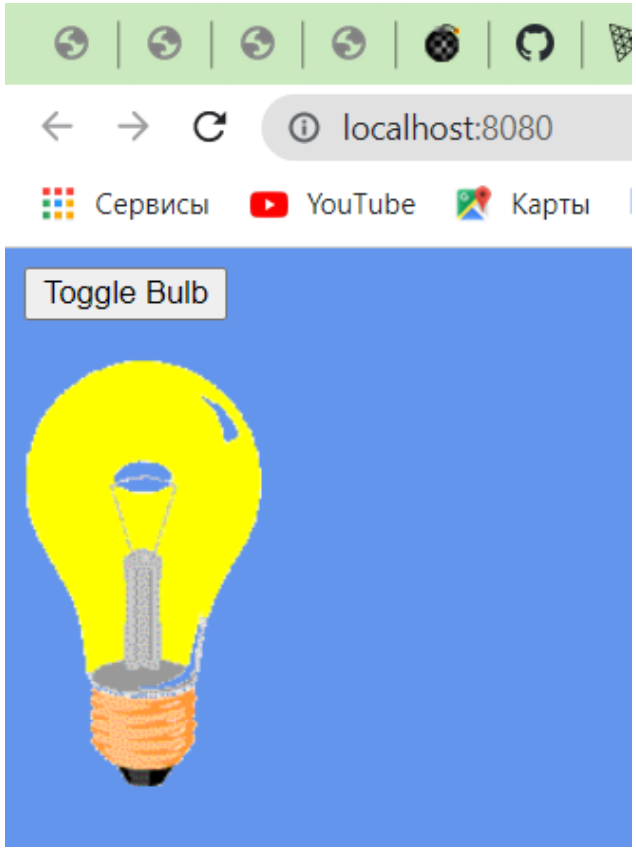
Изменение атрибута элемента *HTML* (2)

```
fun Node.addToggleBulbButton() {
    append {
        div {
            button {
                +"Toggle Bulb"
                type = ButtonType.button
                onClickFunction = { toggleBulb() }
            }
            p {
                img {
                    id = "bulb"
                    src = "pic_bulboff.gif"
                }
            }
        }
    }
}
```

Изменение атрибута элемента *HTML* (3)

```
// ...
            p {
                img {
                    id = "bulb"
                    src = "pic_bulboff.gif"
                }
            }
// ...

var isOn = false

fun toggleBulb() {
    val bulbImage = document.getElementById("bulb") as Image
    bulbImage.src = if (isOn) "pic_bulboff.gif" else "pic_bulbon.gif"
    isOn = !isOn
}
```

Изменение атрибута *HTML* (в браузере)



Изменение атрибута *HTML*. Смена изображения

Изменение стиля элемента *HTML* (1)

При нажатии на кнопку должен изменяться стиль текста

```
fun main() {
    document.bgColor = "6495ED"
    document.fgColor = "yellow"

    window.onload = {
        document.body?.addToggleStyleButton()
    }
}
```

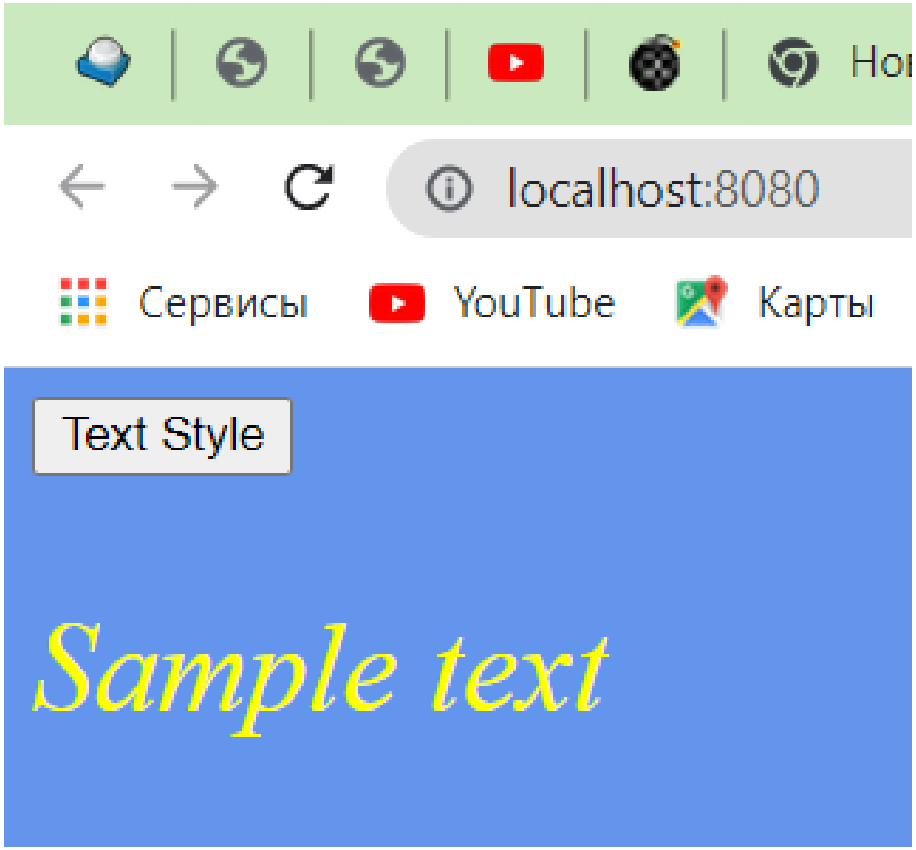
Изменение стиля элемента *HTML* (2)

```
fun Node.addToggleStyleButton() {
    append {
        div {
            button {
                +"Text Style"
                type = ButtonType.button
                onClickFunction = { toggleStyle() }
            }
            p { id = "demo"
                +"Sample text " }
        }
    }
}
```

Изменение стиля элемента *HTML* (3)

```
var isItalic = false

fun Node.toggleStyle() {
    val text = document.getElementById("demo") as
HTMLParagraphElement
    text.style.fontSize = "35px"
    text.style.fontStyle = if (isItalic) "italic" else "normal"
    isItalic = !isItalic
}
```

Изменение стиля *HTML* (в браузере)



Изменение стиля *HTML*

События от мыши (1)

При нажатии на кнопку мыши должен изменяться цвет текста и его фона

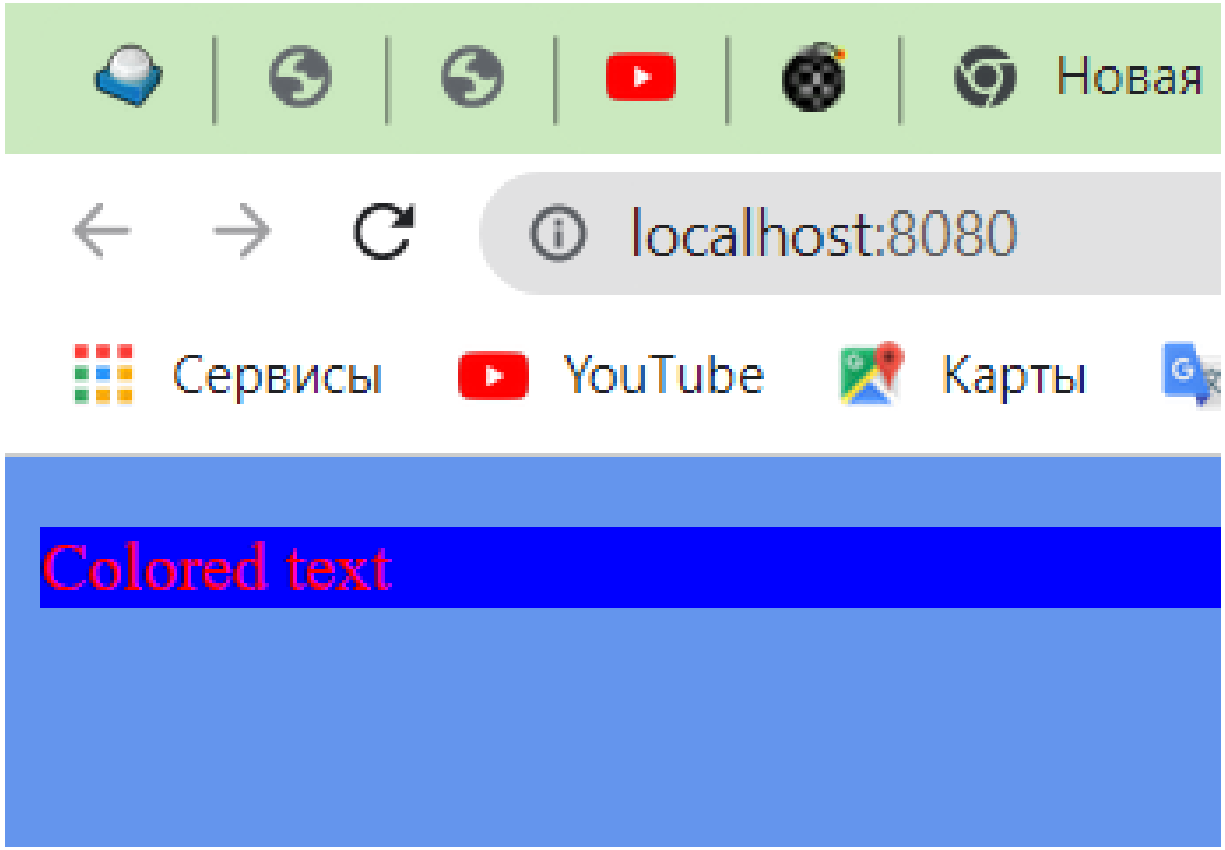
```
fun Node.addMouseDownButton() {
    append {
        p {
            +"Colored text "
            id = "demo"
            onMouseDownFunction = { paint(id, "red", "blue") }
            onMouseUpFunction = { paint(id, "green", "yellow") }
        }
    }
}
```

События от мыши (2)

При нажатии на кнопку мыши должен изменяться цвет текста и его фона

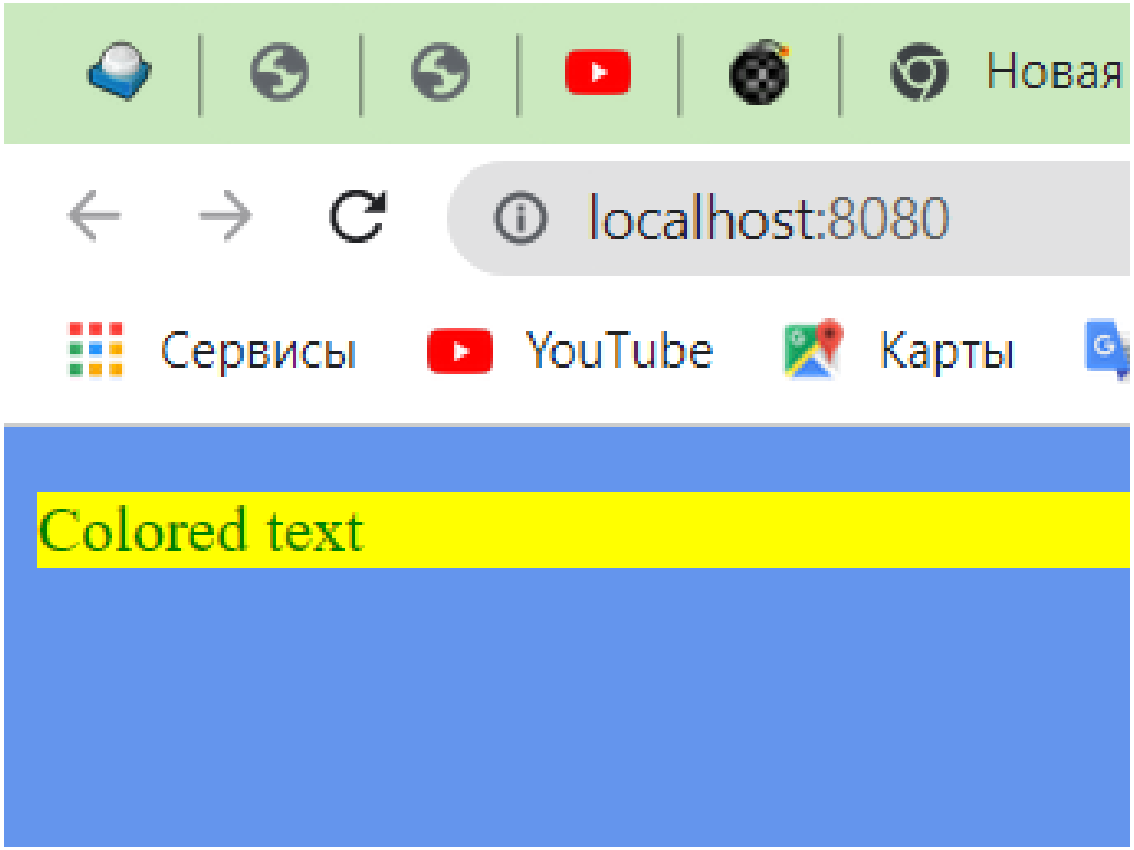
```
fun paint(id: String, color: String, background: String = "White") {
    val text = document.getElementById(id)
        as HTMLParagraphElement
    text.style.color = color
    text.style.background = background
}
```

События от мыши (нажатие)



События от мыши (нажатие)

События от мыши (отпускание)



События от мыши (отпускание)

Выбор элемента мышью(1)

При нажатии на кнопку мыши выводятся имя тега и идентификатор элемента

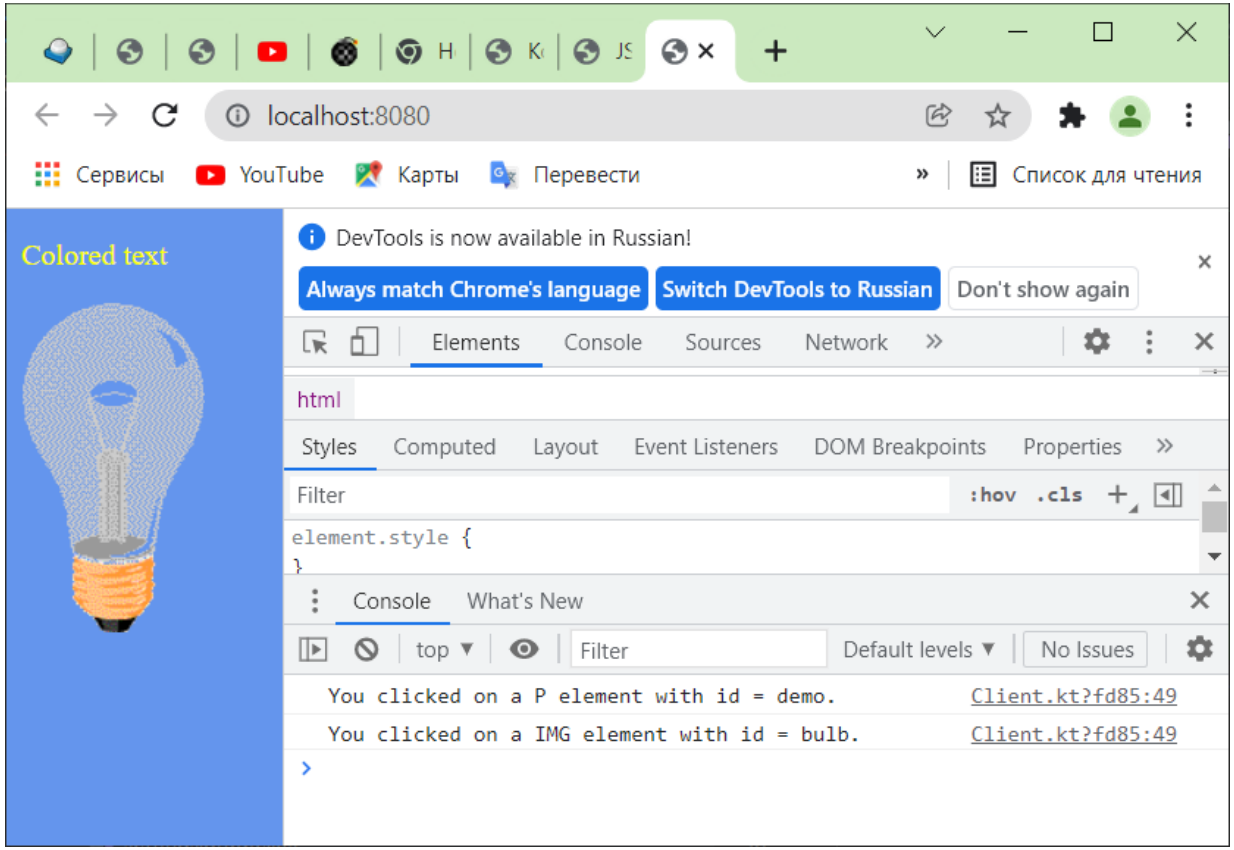
```
fun HTMLElement.addMouseSelectButton() {
    onmousedown = { mouseSelection(it) }
    append {
        p {
            +"Colored text "
            id = "demo"
        }
        img {
            id = "bulb"
            src = "pic_bulboff.gif"
        }
    }
}
```

Выбор элемента мышью(2)

При нажатии на кнопку мыши выводятся имя тега и идентификатор элемента

```
fun mouseSelection(e: MouseEvent) {
    val target: EventTarget = e.target ?: return

    when (target) {
        is HTMLElement -> {
            val tagName = target.tagName
            val id = target.id
            console.info("You clicked on a $tagName element with id =
$id.");
        }
    }
}
```

Вывод выбора элемента мышью на консоль



При нажатии на кнопку мыши выводятся имя тега и идентификатор элемента

Ввод с клавиатуры (1)

При нажатии на кнопку клавиатуры выводятся символ на кнопке и ее код

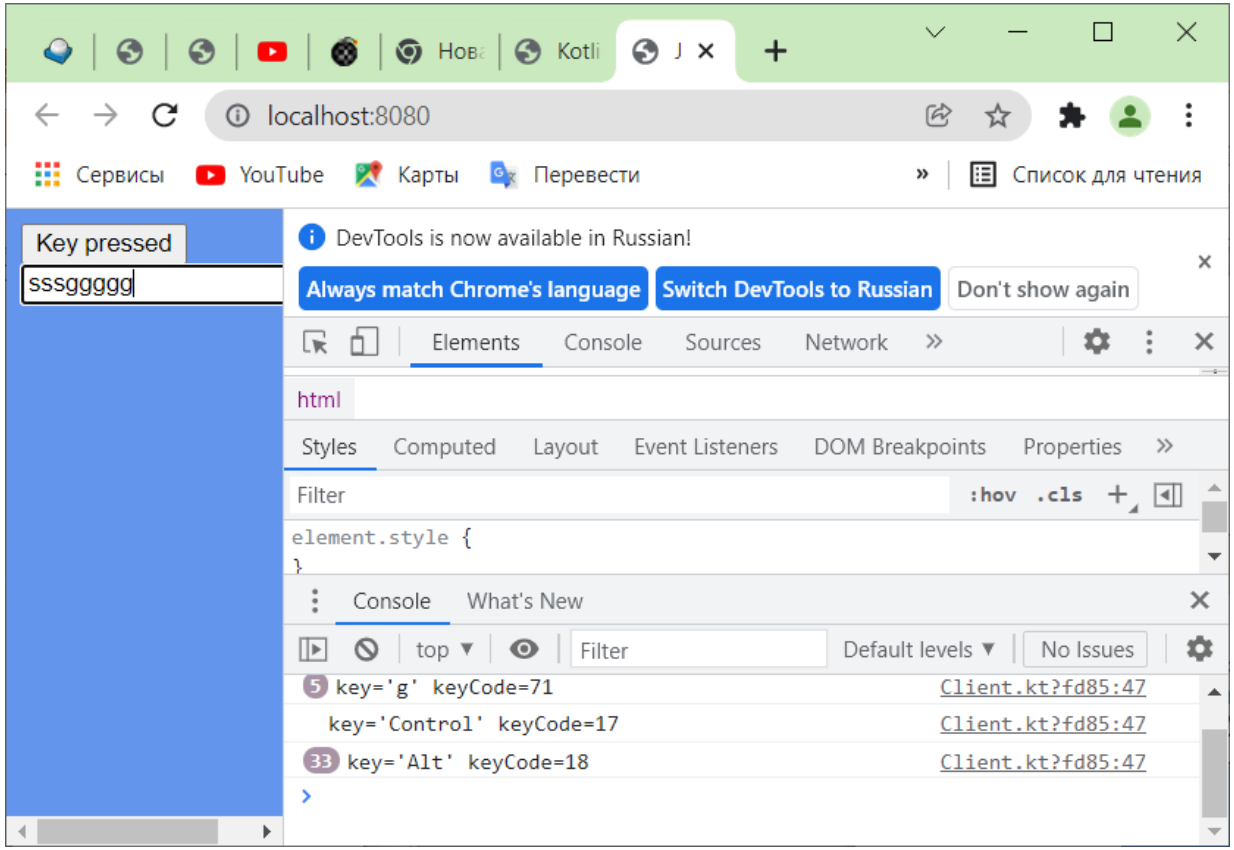
```
fun HTMLElement.addKeyPressedButton() {
    append {
        div {
            button {
                +"Key pressed"
                type = ButtonType.button
                onClickFunction = { toggleBulb() }
            }
            input {
                onkeydown = { keyPressed(this, it) }
            }
        }
    }
}
```

Ввод с клавиатуры (2)

При нажатии на кнопку клавиатуры выводятся символ на кнопке и ее код

```
fun keyPressed(p: INPUT, e: KeyboardEvent) {
    val s = "key='${e.key}' keyCode=${e.keyCode}"
    console.info(s)
}
```

Ввод с клавиатуры (3)



При нажатии на кнопку клавиатуры выводятся символ на кнопке и ее код

6.2 WEB ПРИЛОЖЕНИЕ С ИСПОЛЬЗОВАНИЕМ REACT AND KOTLIN/JS

Визуализация просмотренных и не просмотренных видео

```
data class Video(
    val id: Int,
    val title: String,
    val speaker: String,
    val videoUrl: String
)
val unwatchedVideos = listOf(
    Video(1, "Building and breaking things", "John Doe",
"https://youtu.be/PsaFVLr8t4E"),
    Video(2, "The development process", "Jane Smith",
"https://youtu.be/PsaFVLr8t4E"),
    Video(3, "The Web 7.0", "Matt Miller",
"https://youtu.be/PsaFVLr8t4E")
)
val watchedVideos = listOf(
    Video(4, "Mouseless development", "Tom Jerry",
"https://youtu.be/PsaFVLr8t4E")
)
```

Структура исходной HTML страницы

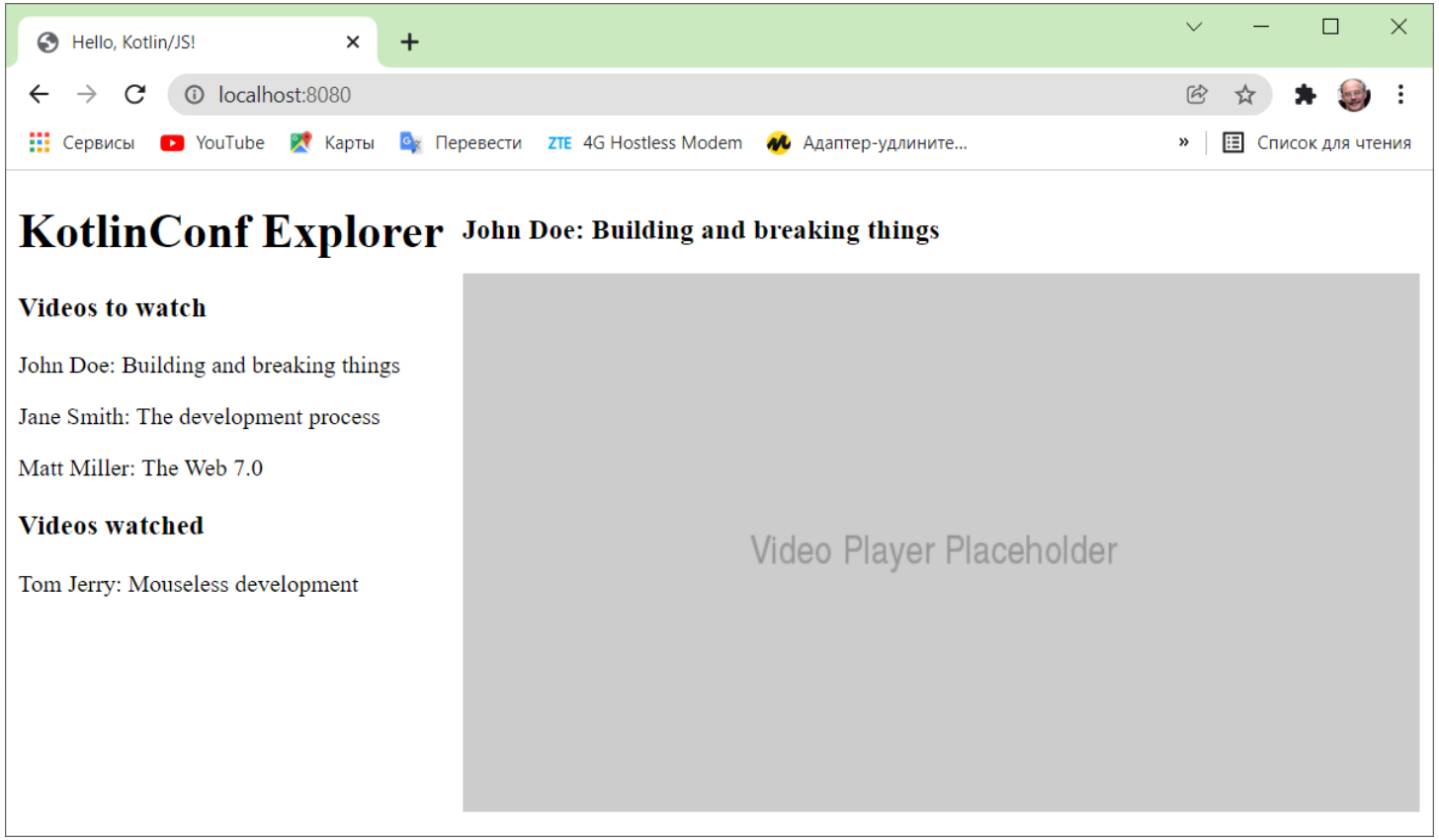
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello, Kotlin/JS!</title>
</head>
<body>
    <div id="root"></div>
    <script src="confexplorer.js"></script>
</body>
</html>
```

Формирование HTML страницы

Структура формируемой страницы:

```
render(document.getElementById("root")) {
    h1 {
        +"KotlinConf Explorer"
    }
    div {
// ...
    }
    styledDiv {
// ...
    }
}
```

Сформированная HTML страница



При нажатии на кнопку клавиатуры выводятся символ на кнопке и ее код

Список видео

```
div {
    h3 {
        +"Videos to watch"
    }
    for(video in unwatchedVideos) {
        p {
            +"${video.speaker}: ${video.title}"
        }
    }
    h3 {
        +"Videos watched"
    }
    for(video in watchedVideos) {
        p {
            +"${video.speaker}: ${video.title}"
        }
    }
}
```

Подключение библиотеки Kotlin Styled в файле *build.gradle.kts*

```
dependencies {
    //...
    // Kotlin Styled (шаг 3)
    implementation("org.jetbrains:kotlin-styled:5.2.1-pre.148-kotlin-1.4.21")
    implementation(npm("styled-components", "~5.2.1"))
    //...
}
```

Импорт библиотек

```
import kotlinx.css.*
import styled.*
```

Выбранное видео

```
styledDiv {
    css {
        position = Position.absolute
        top = 10.px
        right = 10.px
    }
    h3 {
        +"John Doe: Building and breaking things"
    }
    img {
        attrs {
          src =
"https://via.placeholder.com/640x360.png?text=Video+Player+Placeholder"
        }
    }
```

Выделение корневого компонента. Файл **main.kt**

```
fun main() {
    render(document.getElementById("root")) {
        child(app)
    }
}
```

Выделение корневого компонента. Файл **App.kt**

```
val app = fc<Props> {
    h1 {
        +"KotlinConf Explorer"
    }
    div {
// ...
    }
    styledDiv {
// ...
    }
}
```

Проблема

- Компонент приложения не контролирует контент, отображаемый компонентом **videoList**.
- Он жестко запрограммирован, поэтому мы видим один и тот же список дважды.
- Нужен механизм для передачи списка в компонент.

Решение

- Нужно иметь возможность заполнять компонент **videoList** различными типами контента.
- нужно иметь возможность передавать список элементов в качестве атрибута компоненту.
- Добавим атрибут, содержащую список отображаемых видео.
- Определим внешний интерфейс **VideoListProps**, содержащий все атрибуты, которые могут быть переданы компоненту videoList.

Выделение компонента-списка (1) VideoList.kt

```
external interface VideoListProps : Props {
    var videos: List<Video>
}
```

Выделение компонента-списка (2) VideoList.kt

```
val videoList = fc<VideoListProps> { props ->
    var selectedVideo: Video? by useState(null)
    for (video in props.videos) {
        p {
            key = video.id.toString()
            attrs {
                onClickFunction = {
                    selectedVideo = video
                }
            }
            if (video == selectedVideo) {
                +"▶ "
            }
            +"${video.speaker}: ${video.title}"
        }
    }
}
```

Запись списка видео в HTML

```
div {
    h3 {
        +"Videos to watch"
    }
    for(video in unwatchedVideos) {
        p {
            +"${video.speaker}: ${video.title}"
        }
    }
}
```

Запись списка видео в HTML (child)

```
div {
    h3 {
        +"Videos to watch"
    }
    child(videoList) {
        attrs {
            videos = unwatchedVideos
        }
    }
}
```

6.3 РАЗРАБОТКА WEB ПРИЛОЖЕНИЙ НА ЯЗЫКЕ KOTLIN

Ссылки

- **Building Web Applications with React and Kotlin/JS**
  link
- **Creating an interactive website with Ktor**
  link
- **Building a Full Stack Web App with Kotlin Multiplatform**
  link