امروز بهتون یادمیدم که چطور بتونین با DOM از داخل برنامه جاوااسکریپتتون کارکنین.

اگه یادتون باشه یک برانامه نوشتیم که کنسول رو صدا میزد و “Hello, World” رو داخلش چاپ میکرد و همینطور یک فایل index.html ساختیم که بتونیم نتیجه رو نمایش بدیم. خب حالا به قسمتی رسیدیم که میخوایم با DOM ارتباط برقرار کنیم. قبل از این که داخل Main.kt کدی بزنیم لازمه که یک سری تغییرات توی فایل index.html بدیم:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Working With DOM</title>  
</head>  
<body>  
 <div id="root"></div>  
  
 <script src="out/production/KotlinJSHelloWorld/lib/kotlin.js"></script>  
 <script src="out/production/KotlinJSHelloWorld/KotlinJSHelloWorld.js"></script>  
</body>  
</html>

خب کاری که کردیم درواقع تنها اضافه کردن یک div به فایل html­امون بود که بتونیم از طریق کد جاوااسکریپتمون باهاش ارتباط برقرار کنیم.

حالا میتونیم از طریق کد کاتلین باهاش ارتباط برقرار کنیم. برگردیم به Main.kt و کد زیر رو اضافه کنیم:

*document*.getElementById("root")

درواقع کدی که زدیم میاد عنصری که با ای دی root وجود داره رو صدا میزنه که بعدا میتونیم ازش استفاده کنیم.

ولی یه سوال.... اینجا داریم با پاس دادن یه رشته یک عنصر رو صدا میزنیم.اگه اشتباه تایپی داشته باشیم و یا اصا به هر دلیلی یک عنصر null رو صدا بزنیم چه اتفاقی میفته.

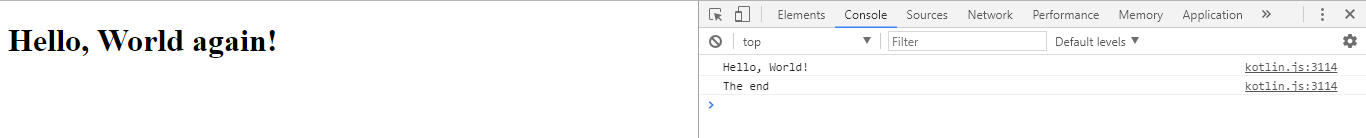
درواقع موقع صدازندش اتفاقی نمیفته، وقع استفاده کردنش منجر به بروز اتفاق میشه.مثلا کد زیر رو نگاه کنین:

val root = *document*.getElementById("root1")  
root.innerHTML = "<h1>Hello, World again!</h1>"

توی کاتلین اگه همچین کدی رو بنویسین، کامپایلر توی خط دوم بهتون ارور میده! دلیلش هم اینه کا کاتلین امنه! کاتلین(اگه درست ازش استفاده کنین) امنه! اجازه نمیده از عنصری که ممکنه null باشه استفاده کنین. برای حل این مشکل چندین راه حل دارین. اولیش اینه که از روش قدیمی استفاده کنیم، یعنی :

fun main(args: Array<String>) {  
 *println*("Hello, World!")  
  
 val root = *document*.getElementById("root")  
 if (root != null) {  
 root.innerHTML = "<h1>Hello, World again!</h1>"  
 }  
  
 *println*("The end")  
}

الان اگه این کد رو بیلد کنین، نتیجه رو توی مرورگر اینجوری میبینین:

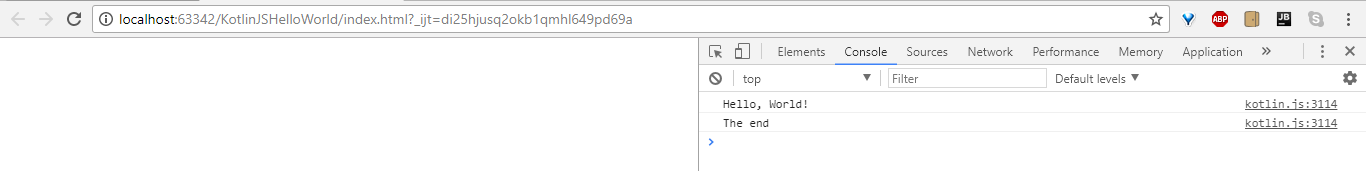


اگه root به root1 تغییر کنه، یعنی مثلا کد زیر:

val root = *document*.getElementById("root1")

نتیجه اینجوری ظاهر میشه:

همینطور که میبینین چون عنصری رو صدا زدیم که null بود مقدارش چاپ نشد. ولی این خوبه اگه یادمون باشه که قبلش null رو چک کنیم و دقیقا زمان هایی به وجود میاد که یادمون میره همچین کاری انجام بدیم.

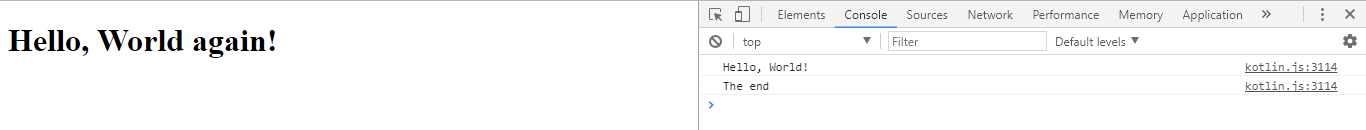


راه حل دیگه استفاده از عملگر "!!" هه. درواقع این عملگر درکاتلین اومده که تنها ارور در زمان کامپایل رو از بین ببره و در واقع توسعه دهنده با صدازدن همچین عملگری به کامپایلر میگه: " تو کاریت نباشه، من میدونم چیکار دارم میکنم". بیاین ازش استفاده کنیم و نتیجه هارو ببینیم :

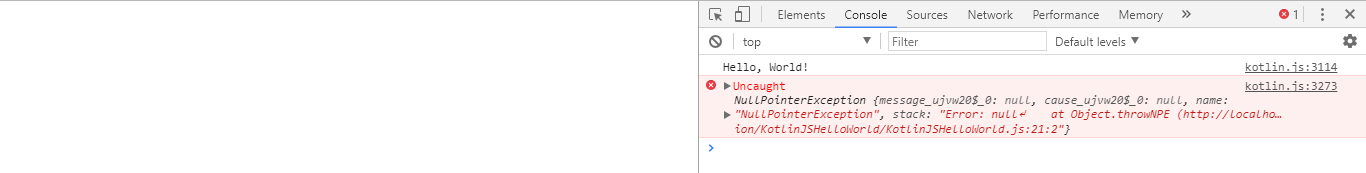
fun main(args: Array<String>) {  
 *println*("Hello, World!")  
  
 val root = *document*.getElementById("root")  
 root!!.innerHTML = "<h1>Hello, World again!</h1>"  
  
 *println*("The end")  
}

و نتیجه :

همینطور که میبینین وقتی عنصر درستی رو صدا بزنیم، برنامه درست کار میکنه. ولی وقتی root1 رو صدا بزنیم:



متوجه میشیم که به NullPointerException میخوریم و متاسفانه این مسئله خیلی از موقع ها پیش میاد و باعث بروز مشکل در سیستم میشه. به همین خاطر به این ارور ها، ارورهای میلیون دلاری هم گفته میشه.



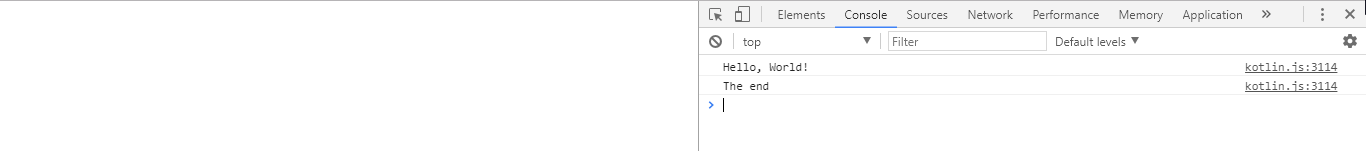
ولی راه حل درستی که کاتلین پیشنهاد میکنه استفاده از عملگر "?" هه . کد زیر رو نگاه کنین:

fun main(args: Array<String>) {  
 *println*("Hello, World!")  
  
 val root = *document*.getElementById("root")  
 root?.innerHTML = "<h1>Hello, World again!</h1>"  
  
 *println*("The end")  
}

در واقع وقتی کامپایلر به اون نقطه میرسه نگاه میکنه، اگه عنصری که داره ازش استفاده میشه null نبود، عمل لازم رو روش انجام میده، ولی اگر null بود از روی اون خط رد میشه.

خب معلومه که اگه این کد رو اجرا کنیم نتیجه درست و دلخواهمون رو میگیریم.بذارین با root1 اجرا کنیم و نتیجه رو ببینیم:

خب همینطور که میبنین، همون نتیجه که میخواستیم رو گرفتیم و علاوه بر اون چون کامپایلر بهمون موقع کدنویسی ارور میده، خودمون قبل از اجرا متوجه میشیم و با استفاده از این عملگر از اجراش جلوگیری میکنیم.

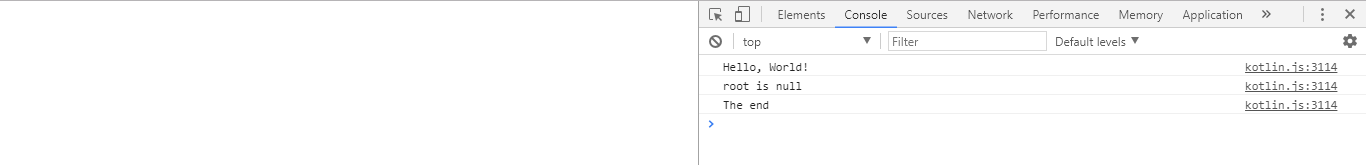


حالا اگه توسعه­دهنده که ما باشیم بخواد اگه null دریافت کرد کار دیگه ای رو انجام بده(مثلا موقعی که null دریافت کرد یک رشته رو چاپ کنه) اون موقع باید چیکار کنه. کاتلین اینجا هم راه حل داره. کد زیر رو نگاه کنین:

fun main(args: Array<String>) {  
 *println*("Hello, World!")  
  
 val root = *document*.getElementById("root1")  
 *println*("root is $root")  
 root?.innerHTML = "<h1>Hello, World again!</h1>"  
  
 *println*("The end")  
}

در واقع یکی از خوبیای کاتلین اینه! کاتلین وقتی یک یک شی null رو برای print دریافت میکنه، مقدار null رو هم چاپ میکنه.یعنی میتونین نه تنها ارور error نمیده، بلکه میتونین ازش استفاده کنین.الان اگه این کد رو اجرا کنین نتیجه زیر رو میبینین:

همینطور که میبینین مقدار null رو چاپ میکنه.

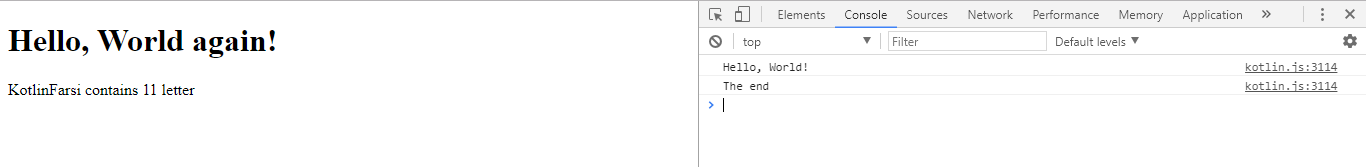


حالا بیاین کد html ای که میخوایم به div اضافه کنیم رو یکم مشکل­تر کنیم، یک کد چندخطی استفاده کنیم.

fun main(args: Array<String>) {  
 *println*("Hello, World!")  
  
 val message = "KotlinFarsi"  
  
 val html = """  
 <h1>Hello, World again!</h1>  
 <p>KotlinFarsi contains ${message.length} letters</p>  
 """  
 val root = *document*.getElementById("root")  
 root?.innerHTML = html  
  
 *println*("The end")  
}

توی این کد چندتا نکته جدید داریم. یکی این که از یک متغییر برای نوشتن کد چندخطی html استفاده کردیم و برای این که به کامپایلر بفهمونیم کدمون چندخطیه، کد html رو داخل یک جفت """ قرار دادیم. و همینطور فهمیدیم که اگه بخوایم میتونیم داخل رشته هم از مقدار یک متغیر استفاده کنیم. درواقع زمانی که از ${} استفاده میکنیم، کامپایلر هرچی داخل اکولادها بنویسین رو به عنوان یک کد حساب میکنه، نه رشته، و در انتها مقدار اون کدی که نوشتیم رو به صورت رشته چاپ میکنه.

خروجی کد بالا :



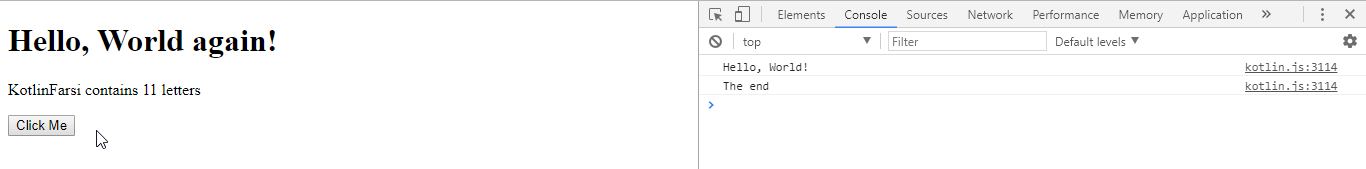
حالا بیاین یک دکمه به html اضافه کنیم. دکمه ای که وقتی روش کلیک شد یک عبارت رو چاپ کنه:

fun main(args: Array<String>) {  
 *println*("Hello, World!")  
  
 val message = "KotlinFarsi"  
  
 val html = """  
 <h1>Hello, World again!</h1>  
 <p>KotlinFarsi contains ${message.length} letters</p>  
 <button id="btn">Click Me</button>  
 """  
 val root = *document*.getElementById("root")  
 root?.innerHTML = html  
 val btn = *document*.getElementById("btn")  
 btn?.addEventListener("click",**{** *println*("Clicked") **}**)  
  
 *println*("The end")  
}

اومدیم به متغییر html یک دکمه اضافه کردیم و توی کدمون یک EventListener و زمانی که برروی دکمه کلیک میشه “Clicked” توی کنسول به نمایش در میاد.

قبل از فشار دادن دکمه:

و بعد از فشار دادن دکمه :



همینطور که میبینین وقتی دکمه رو فشار دادیم Clicked چاپ شد.

