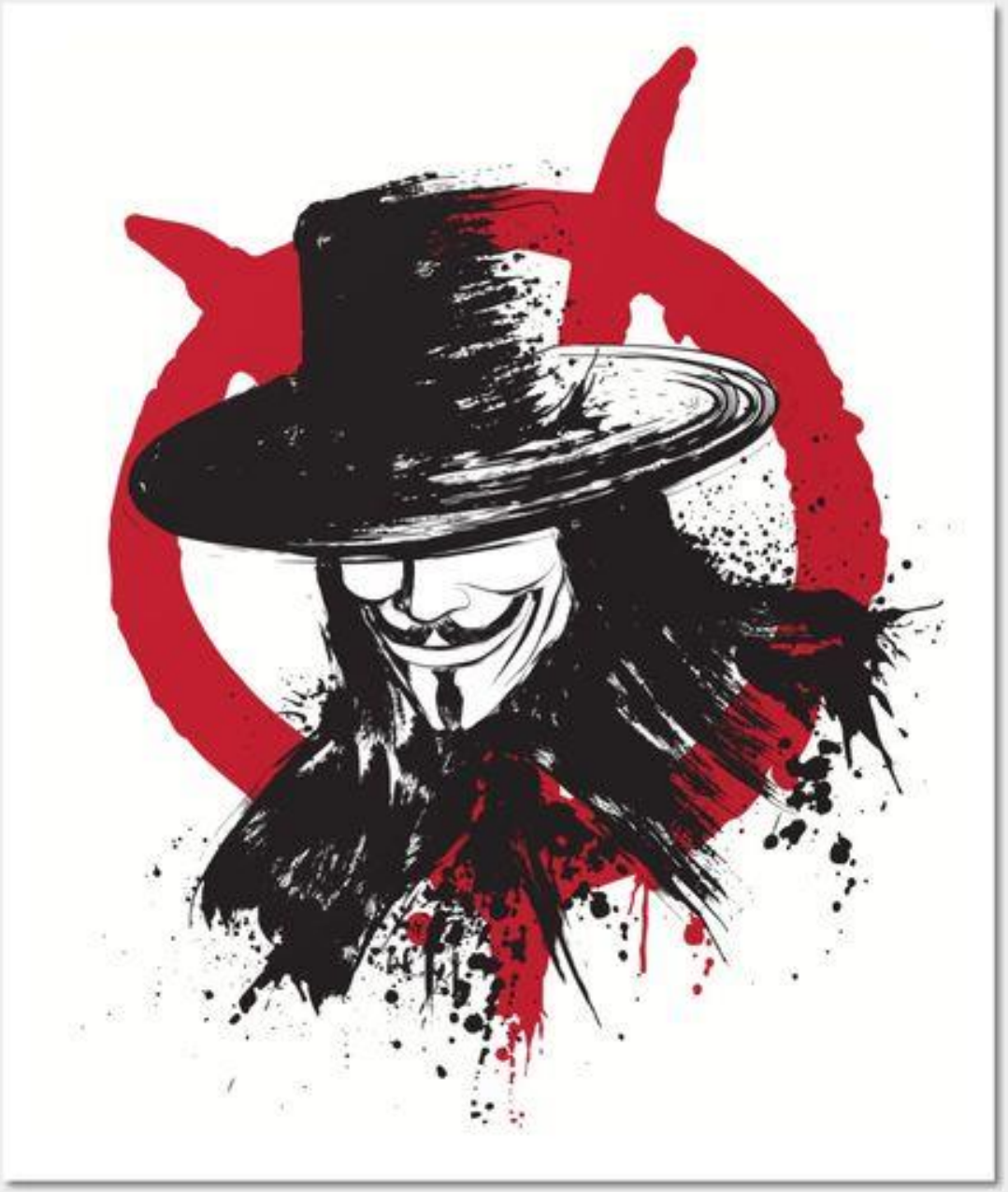
Web Application Penetration Testing



First Session Note

By TheSecDude

زبان برنامه نویسی جاوااسکریپت ؟ یک زبان برنامه نویسی و اسکریپت نویسی است که به شما اجازه میدهد که چیز ها و امکاناتی پیچیده را بر روی صفحات وب پیاده‌سازی کنید . هر زمانی که صفحات وب از حالت Static خارج می‌شوند و حالتی پویا به خود میگیرند مثلاً زمانی که دادهها اپدیت میشوند، المنت هایی مثل نقشه، انیمیشن های D, ۲D۳ و ... دارند میتوان گفت که جاوااسکریپت در آن صفحه دخیل است .
امروزه از این زبان برنامه نویسی که یک روز فقط برای صفحات وب استفاده میشد استفادههای زیادتری وجود دارد . مثلاً در نوشتن نرم افزارهای دسکتاپ، نوشتن نرم افزارهای اندرویدی، نوشتن Back-End سایتها و حتی در حوزه امنیت نوشتن اکسپلویت های اندرویدی استفاده می‌شود . این زبان به مانند زبانی مثل پایتون استفادههای زیادی دارد و یادگیری آن برای هر شخصی که میخواهد در حوزه تست نفوذ وب کار کند الزامیست چرا که بسیاری از حفرات امنیتی در JavaScript سایتها وجود دارد .
پدر زبان جاوااسکریپت Ecma Script است که امکانات جاوااسکریپت از آن نشعت گرفته است . هر جایی که Engine اجرا کننده جاوااسکریپت وجود داشته باشد کد های Ecma Script نیز قابل اجراست .

از جاوااسکریپت در کد های صفحات وب به دوشکل استفاده می‌شود :

۱. از طریق تگ script در کدهای HTML صفحه
۲. از طرق یک فایل جداگانه که به صفحه از طریق تگ script متصل می‌شود .

[۰۰:۱۶:۴۷]
تگ script ؟ در HTML تگهای زیادی وجود دارد که هر کدام کاری را انجام میدهد . اگر به کدهای یک صفحه HTML دقت کنید تگها را که مابین <> قرار دارند خواهید دید بالای ۹۰ درصد از تگهای HTML وقتی باز می‌شوند <tag> بایستی بسته نیز شوند یعنی به شکل </tag> . یکی از این تگها <script> است که تگ بسته آن به شکل </script> می‌باشد . ما بین این تگ کدهای جاوا اسکریپت صفحه مورد نظر ما قرار میگیرد که توسط مرورگر اجرا می‌شوند .

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<script>
    alert("hello World!")
</script>

</body>
</html>
```

این یکی از روشهای وارد کردن کدهای جاوااسکریپت در صفحات وب است .

لینک کردن فایل JS در صفحات وب ؟ علاوه بر اینکه میتوانید از طریق تگ script کدهای جاوااسکریپت خود را مستقیماً وارد صفحه کنیم میتوانیم آن‌ها را در یک فایل جدا با پسوند .js قرار دهیم و سپس آن‌ها را به صفحه لینک نماییم . برای اینکار یک فایل جاوااسکریپت در در کنار صفحه وب یا در هر جایی ایجاد میکنیم و سپس از طریق تگ script به شکل زیر آن را به صفحه خود لینک مینماییم .

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

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

</body>
</html>
```

برای اینکه فایل خود را به تگ script بدهیم باید از attribute خاص آن یعنی src استفاده کنیم . Path فایل js. خود را به src میدهیم که در تصویر بالا یعنی یک دایرکتوری عقب، در دایرکتوری assets، فایل script.js .

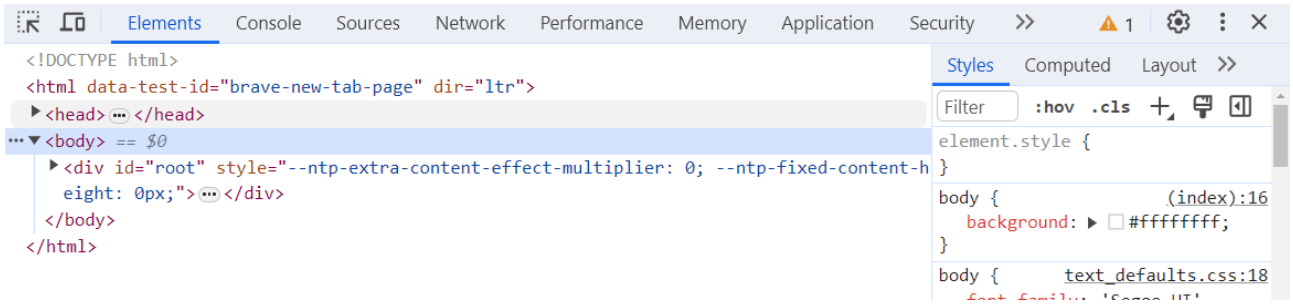
متغیر ها یا Variables در جاوااسکریپت ؟ متغیر ها را همیشه هر کسی که میخواهد تعریف کند آن را به یک ظرفی تشبیه میکند که میتواند چیزی (مقداری) را در خود نگهداری کند و هر وقت که خواستیم به آن چیز (مقدار) اشاره کنیم کافیست که به ظرف اشاره نماییم . هر متغیر یک نامی دارد که به مقدار درون آن اشاره میکند . در هر زبان برنامه نویسی متغیرها وجود مفهوم یکسانی دارند ولی ممکن است نحوه تعریف کردن آن‌ها متفاوت باشد . در جاوااسکریپت متغیر ها با کلماتی کلیدی تعریف می‌شوند مانند let, var, const . تفاوتهایی که دارند زیاد مهم نیستند و فقط نحوه تعریف آن‌هاست که اهمیت دارند . اگر بخواهیم به تفاوتها اشاره کنیم میتوانیم بگوییم که برخی از آن‌ها در حالت global کاربرد دارند، برخی فقط در توابع تعریف می‌شوند و ... .

```
<script>
    let variable_name1 = value;
    var varialbe_name2 = value;
    const variable_name3 = value;
</script>
```

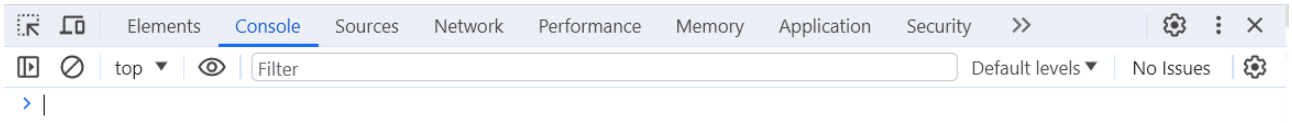
در بالا سه متغیر با نامهای variable_name۱, variable_name۲, variable_name۳ تعریف کردیم و value را در هر سه قرار دادیم . اگر بخواهیم به صورت واقعی مقدار دهی کنیم میتوانیم به شکل زیر عمل نماییم :

```
<script>
    let name = "Alex";
    var age = 24;
    const height = 1.82;
</script>
```

console.log در جاوااسکریپت چه میکند ؟ جاوااسکریپت یکی از اجزای جداناشدنی صفحات وب و همچنین مرورگهاست . در مرورگرها صفحه‌ای وجود دارد تحت عنوان Developer tools که با کلید F۱۲ یا Ctrl+Shift+I باز می‌شود . در هر Tab از مرورگر میتوانیم آن صفحه را باز کنید و آن صفحه مختص آن tab خواهد بود و دسترسی به tab های دیگر ندارد .

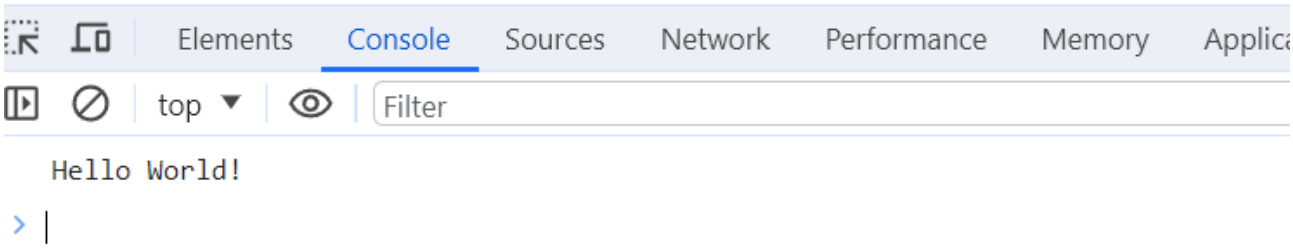


در این پنجره اطلاعات بسیار زیاد و کاربردی درمورد tab مورد نظر وجود دارد . اگر به دکمه های بالای این صفحه نگاه کنیم دکمه Console را میبینیم که اگر بر روی آن کلیک کنیم صفحه‌ای برای ما باز می‌شود به شکل زیر :



در صفحه Console ما میتوانیم کد های جاوااسکریپت خود را به صفحه tab خودمان تزریق کنیم و واکنش صفحه نسبت به آن کد را ببینیم . هر کد جاوااسکریپت در این صفحه قابل اجراست و در ادامه بسیار کارا خواهد بود . console.log اشاره میکند به این صفحه و هر چیزی را که به این دستور بدهیم در این صفحه برای ما مینویسد . مثلاً اگر بنویسیم (“Hello World!”)console.log برای ما جمله Hello World! را در این صفحه می‌نویسد .

```
<script>
    console.log("Hello World!")
</script>
```



Operation ها در جاوااسکریپت ؟ زبان‌های برنامه نویسی بر گرفته از ریاضیات هستند که به زبان کامپیوتر تبدیل شده‌اند . به همین خاطر در زبان برنامه نویسی عملیات هایی که در ریاضیات انجام می‌شوند را نیز داریم . مثلاً عمل جمع، تفریق، تقسیم، باقی مانده، ضرب، توان , ... حال آن‌ها را با هم بررسی کنیم .

عمل جمع در جاوااسکریپت ؟ اگر یک متغیر داشته باشیم به نام x_var که مقدار داخل آن ۱۳۷ است و بخواهیم آن را با ۲۵۴ جمع کنیم و مقدار جدید را در x_var قرار دهیم کافیست که از Operation جمع که با علامت + مشخص می‌شود استفاده کنیم :

```
> let x_var = 137
  x_var = x_var + 254
  console.log(x_var)
391
```

علاوه بر روش بالا میتوانیم از += نیز استفاده کنیم که اگر بگوییم x_var += ۲۴۵ به این معناست که مقداری داخل x_var را با ۲۵۴ جمع کن و سپس آن را در خود x_var قرار بده :

```
>     let x_var = 137
      x_var += 254
      console.log(x_var)
391
```

عمل تفریق در جاوااسکریپت ؟ به مانند عمل جمع عمل تفریق نیز داریم که با علامت – انجام می‌شود .

```
> let x_var = 137
> let x_var = 137
  x_var -= 254
  console.log(x_var)
-117
```

عمل تقسیم در جاوااسکریپت ؟ عمل تقسیم با علامت / انجام می‌شود و به مانند جمع و تفریق می‌باشد .

```
> let x var = 137
> let x_var = 137
  x_var /= 254
  console.log(x_var)
0.5393700787401575
```

بدست آوردن باقی‌مانده تقسیم دو مقدار در جاوااسکریپت ؟ برای اینکار کافیست که از ٪ استفاده کنیم و مقدار باقی‌مانده تقسیم دو مقدار بر هم را بدست اوریم :

```
> let x_var = 3
```

```
> let x_var = 3
  x_var %= 2
  console.log(x_var)
1
```

عمل ضرب در جاوااسکریپت ؟ برای انجام عمل ضرب از علامت * استفاده میکنیم و دقیقاً به مانند موارد قبلی می‌باشد :

```
> let x_var = 137
```

```
> let x_var = 137
  x_var *= 254
  console.log(x_var)
34798
```

انجام عمل توان در جاوااسکریپت ؟ برای انجام عمل توان از علامت ** استفاده میکنیم و به حالت زیر است :

```
> let x_var = 12
```

```
> let x_var = 12
  x_var **= 10
  console.log(x_var)
61917364224
```

Operand ها در جاوااسکریپت ؟ در زبان‌های برنامه نویسی به حالت دودیی عمل می‌شود . یعنی یا یک چیز مقدارش true است و یا برابر false می‌باشد . Operand ها جهت قیاس دو مقدار با هم استفاده می‌شوند و در نتیجه قیاس true یا false بر میگردد . از آن‌ها در شرطها استفاده می‌شود .

```
> 42 > 10
true
> 42 < 10
false
> 42 >= 42
true
> 42 <= 42
true
> 42 === 42
true
> 42 === "42"
false
> 42 == 42
true
> 42 == "42"
true
> 42 != 42
false
```

چیزی که در Operand ها نیاز می‌بینم که توضیح دهد تفاوت == و === است . در == نوع داده‌ای که قیاس میکنیم را قیاس نمیکند و به همین خاطر ۴۲ برابر "۴۲" است با وجود اینکه نوع آن‌ها با هم تفاوت دارند ولی === نوع دادهها را نیز قیاس میکند و به همین خاطر ۴۲ === "۴۲" جواب false میدهد .

شرطها در جاوااسکریپت ؟ شروط از مهم‌ترین قسمت‌های برنامه نویسی هستند و میتوان گفت که زبان برنامه نویسی وجود ندارند که عبارات شرطی در آن نباشد . معمولاً در عموم زبان‌های برنامه نویسی برای تعریف یک شرط از کلمه کلید if استفاده می‌شود و جاوااسکریپت هم همینطور است .
کافیست که کلمه if را نوشته و در پرانتز های جلوی آن شرط مورد نظر خود را بنویسیم و سپس در آکولاد هایی که باز میکنیم عباراتی را بنویسیم که در صورت درست بودن شرط می بایست اجرا شوند :

```
> if (condition){
      // Statement1 ...
      // Statement2 ...
      // Statement3 ...
  }
```

مثلاً به مثال زیر توجه کنید :

```
let a = 20
if (a > 10){
    console.log("a is bigger than 10")
}
```

```
a is bigger than 10
```

یا هم به شرط زیر نگاه کنید که به علت غلط بودن عبارت شرطی آن اجرا نشده است :

```
let a = 20
if (a < 10){
    console.log("a is smaller than 10")
}
```

علاوه بر یک شرط ما میتوانیم چندین شرط را نیز تعریف کنیم که در صورت درست بودن هر کدام عبارات موجود در آن اجرا می‌شوند . برای اینکار از if و else if با هم استفاده میکنیم :

```
let a = 20
if (a > 24){
    console.log("a is bigger than 24")
}else if (a > 23){
    console.log("a is bigger than 23")
}else if (a > 22){
    console.log("a is bigger than 22")
}else if (a > 19){
    console.log("a is bigger than 19")
}
```

```
a is bigger than 19
```

اگر هم بخواهیم در انتهای شروط خود چیزی را بنویسیم که در صورت درست نبود هیچ کدام از شرط ها آن عبارات اجرا شوند کافیست که از دستور else استفاده کنیم که درصورت درست نبودن هیچکدام از شروطمان در نهایت عبارات داخل آن اجرا می‌شوند :

```
let a = 20
if (a > 24){
    console.log("a is bigger than 24")
}else if (a > 23){
    console.log("a is bigger than 23")
}else if (a > 22){
    console.log("a is bigger than 22")
}else if (a > 21){
    console.log("a is bigger than 21")
}else {
    console.log("a is not bigger than 23, 22, 21")
}
```

```
a is not bigger than 23, 22, 21
```

میتوانیم در هر if چندین شرط بنویسیم که با && به معنی "و" و || به معنی "یا" از هم جدا می‌شوند . مثلاً در مثال زیر گفتیم که اگر a بزرگ‌تر از ۲۵ بود و a کوچکتر از ۳۰ بود شرط اجرا شود، در قیاس دو داده با && هر دو باید درست باشند :

```
let a = 27
if (a > 25 && a < 30){
    console.log("a is bigger than 25 and smaller than 30")
}
```

```
a is bigger than 25 and smaller than 30
```

یا در مثال زیر از || به معنی یا استفاده کرده‌ایم که در صورت درست بودن هر کدام از شرطها if اجرا می‌شود و نیازی نیست هر دو درست باشند :

```
let a = 27
if (a > 25 || a < 21){
    console.log("a is bigger than 25 or smaller than 21")
}
```

```
a is bigger than 25 or smaller than 21
```

همچنین میتوانیم از طریق پرانتز و || و && ترکیبی از چندین شرط را ایجاد کنیم و دستورات شرطی ترکیبی بسازیم، به شکل زیر :

```
if ((condition1 && condition2) || (condition3 || condition4) && (condition5 || condition6)){
    // Statement1 ...
    // Statement2 ...
    // Statement3 ...
}
```

در شروط، ابتدا شرطهای داخل داخلی ترین پرانتز ها بررسی می‌شوند و سپس جواب‌ها با جواب‌های پرانتز های دیگر ... مثل عملیات ها در ریاضیات که از پرانتز شروع می‌شوند .

String ها یا رشته‌ها در جاوااسکریپت ؟ در زبان‌های سطح بالا مثل پایتون، جاوااسکریپت، PHP و ... یکی از انواع متغیر ها رشته‌ها یا String ها هستند . نوع یک متغیر بر اساس مقداری که در آن ریخته می‌شود تعیین می‌شود و به همین اساس اگر مقدار یک متغیر مابین "" یا '' یا `` قرار بگیرند، آن متغیر String نامیده می‌شود . برای مثال :

```
let first_name = "Ahmed"
let last_name = 'Teymouri'
let username = `@TheSecdude`

console.log(typeof first_name)
console.log(typeof last_name)
console.log(typeof username)
```

```
string
string
string
```

میبینید که سه متغیر تعریف کردیم که هر سه مقدار String دارند و از طریق دستور typeof که نام متغیر را جلوی آن مینویسیم نوع آن را مشخص کردیم . یک Attribute در String ها وجود دارد به نام length که در جواب برای ما اندازه هر String را بر میگرداند که میتوانید آن را در هر رشته ای ببینید :

اگر

```
let first_name = "Ahmed"

console.log(first_name.length)
```

```
5
```

بخواهیم که چند متغیر رشته ای را به هم بچسبانیم یا به عبارت دیگر آن‌ها را Concatenate کنیم کافیست که از عملگر + استفاده کنیم، مثلاً در مثال زیر دو کلمه Hello و Word که به ترتیب در متغیر های x و y بودند را با + به هم چسباندیم و در متغیر z قرار دادیم :

```
let x = "Hello"
let y = "World"

let z = x + y
console.log(z)
```

```
HelloWorld
```

دیدید که از طریق "" و '' و `` ها توانستیم String ها را تعریف کنیم . "" و '' تفاوت آنچنانی با هم ندارند ولی `` یا BackTick قابلیتی دارند که "" و '' ندارند که به آن Format String میگویند . از طریق این قابلیت ما میتوانیم در String هایمان عملیات هایی را انجام دهیم . مثل میتوانیم مستقیماً نام یک متغیر را بنویسیم تا به جای آن مقدار آن قرار بگیرد بدون اینکه بخواهیم String ها را به هم Concatenate کنیم . کافیست نام متغیر مورد نظر خود را در ${} بنویسیم، به شکل زیر :

```
let username = "@TheSecDude"
let message = `Hello ${username}`

console.log(message)
```

```
Hello @TheSecDude
```

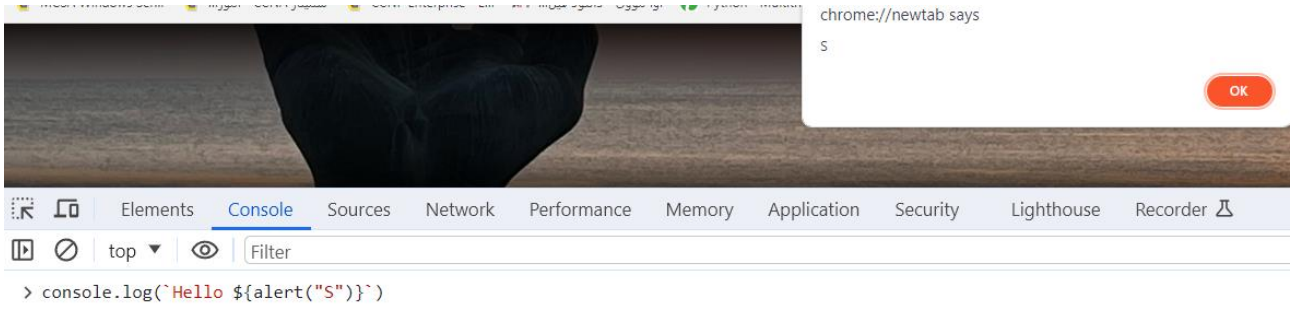
قابلیت Format String تنها به همینجا ختم نمیشود و ما میتوانیم علاوه بر کاری که در بالا انجام دادیم، عملیاتهای ریاضیاتی و ... را نیز از طریق آن انجام دهیم، مثلاً در مثال زیر چند عدد را با هم جمع کردیم :

```
let username = "@TheSecDude"
let points = 24
let bonus = 10
let message = `Hello ${username}, You have ${points + bonus} points .`

console.log(message)
```

```
Hello @TheSecDude, You have 34 points .
```

جناب BackTick در صفحات وب میتواند موجب حفره امنیتی XSS شود و مهاجمین میتوانند از آن سوءاستفاده کنند چرا که از طریق آن میتوانند کدهای خود را اجرا نمایند، در آینده بیشتر و خیلی بیشتر درباره آن صحبت خواهیم کرد :



این رو فعلاً به یاد داشته باشیم که از این مشکل امنیتی میتوانید برای Bypass کردن فایروال های تحت وب استفاده کنیم . آینده بسیار نزدیک است، قطعاً صحبت میکنیم درموردش .

دقت شود که اگر یک بار یک متغیر را با let, var, const تعریف کردید، نمیتوانید دوباره آن را تعریف کنید و فقط میتوانید مقدار آن را عوض کنید :

```
let first_name = "Ahmed"
let first_name = "Javid"
```

```
let first_name = "Ahmed"
first_name = "Javid"
```

آرایه یا لیست ؟ در جاوااسکریپت ! از لحاظ لغوی ارایه با لیست متفاوت است . ارایه به متغیری گفته می‌شود که دسته‌ای از مقدار ها را درخود دارد و همه این مقادیر نوع یکسانی دارند و لیست به متغیر گفته می‌شود که دسته از مقادیر را در خود دارد و این مقادیر میتوانند از انواع مختلفی باشند . اگر با پایتون آشنا باشید قطعاً لیستها را میشناسید . در جاوااسکریپت نیز وجود دارند اما به جای لیست به آن‌ها ارایه یا Array گفته می‌شود در صورتی که ویژگی‌های لیست ها را دارد . نمیدونم شاید اشتباه لغوی باشه ولی خب ارایه ها در جاوااسکریپت میتوانند مقادیر متفاوتی با type متفاوتی داشته باشند . در جاوااسکریپت ارایه را میتوان به سه شکل تعریف کرد که در زیر نمایش داده‌ایم :

```
const array_name = [value_1, value_2, value_3, ...]
const array_name = Array(value_1, value_2, value_3, ...)
const array_1 = [1, 2, "3", 4.5]
const array_2 = Array(1, 2, "3", 4.5)
const array_3 = Array.of(1, 2, "3", 4.5)

console.log(typeof(array_1))
console.log(typeof(array_2))
console.log(typeof(array_3))
```

```
object
object
object
```

و البته جاوااسکریپت نوع آن‌ها را object میداند، همانطوری که در بالا میبینید . البته دو نوع آخر با هم تفاوتهایی دارند که من نمیدونم ولی خب میگن متفاوت اند و مث اینکه فعلاً برامون مهم نیست .
ارایه های چندبعدی یا Multidimensional Array نیز وجود دارند . یعنی میتوانیم ارایه های تودرتو داشته باشیم که مقادیر آن‌ها نیز ارایه باشند، مثلاً به شکل زیر :

```
const array_1 = [
    [1, 2, 3],
    ["a", "b", "c"],
    [
        ["Ahmed", "Alex", "Roz"],
        [27, 25, 26],
        [1.72, 1.95, 1.70]
    ]
]

console.log(array_1)
```

```
▼ (3) [Array(3), Array(3), Array(3)]
  ▶ 0: (3) [1, 2, 3]
  ▶ 1: (3) ['a', 'b', 'c']
  ▼ 2: Array(3)
    ▶ 0: (3) ['Ahmed', 'Alex', 'Roz']
    ▶ 1: (3) [27, 25, 26]
    ▶ 2: (3) [1.72, 1.95, 1.7]
      length: 3
    ▶ [[Prototype]]: Array(0)
    length: 3
  ▶ [[Prototype]]: Array(0)
```

اگر بخواهیم به مقادیر یکی از index های ارایه خود دسترسی بیابیم کافیست که array_name[index_number] را بنویسیم . یعنی نام ارایه و سپس در [] شماره index مورد نظرمان . حال اگر ارایه چندبعدی داشتیم میتوانیم چند [] بنویسیم . مثلاً array_name[index_۱][index_۲] .
به مثال زیر دقت کنید :

```
const matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

console.log(matrix[0])
console.log(matrix[0][1])
```

```
▶ (3) [1, 2, 3]
2
```

در اولین console.log ما به مقادیر داخل index شماره ۰ از ارایه matrix که [۱, ۲, ۳] است دسترسی یافته ایم و در console.log دوم به مقدار داخل index شمار ه۱ از مقدار داخل index شماره ۰ از ارایه matrix که ۲ است دسترسی یافته ایم . یه کم ممکنه پیچیده باشه ولی خب همینه که هست و باید فهمیدش .
اگر یه وقتی بخواهیم اندازه یک ارایه (تعداد index ها) را بفهمیم کافیست از Attribute length که در String ها بود و ارایه ها هم هست استفاده کنیم . مثلاً ارایه زیر سه مقدار را درخود دارد که توسط length نمایش داده شده است :

```
const array_1 = [1, 2, 3]

console.log(array_1.length)
```

```
3
```

متدهایی در ارایهها وجود دارد که بهتر است با آن‌ها آشنا شویم . اولین متد push است که یک مقدار را میگیرد و آن را به انتهای ارایه اضافه میکند :

```
let array_x = [1, 2, 3]
console.log(array_x)
array_x.push("x")
console.log(array_x)
```

```
▶ (3) [1, 2, 3]
▶ (4) [1, 2, 3, 'x']
```

متد بعدی متد pop است که آخرین index را از ارایه حذف میکند :

```
let array_x = [1, 2, 3]
console.log(array_x)
array_x.pop()
console.log(array_x)
```

```
▶ (3) [1, 2, 3]
▶ (2) [1, 2]
```

متد بعدی unshift است که یک مقدار را میگیرد و آن را به ابتدای ارایه اضافه میکند :

```
let array_x = [1, 2, 3]
console.log(array_x)
array_x.unshift("x")
console.log(array_x)
```

```
▶ (3) [1, 2, 3]
▶ (4) ['x', 1, 2, 3]
```

و متد بعدی متد shift است که اگر آن را بر روی یک ارایه اجرا کنید اولین index آن حذف خواهد شد :

```
let array_x = [1, 2, 3]
console.log(array_x)
array_x.shift()
console.log(array_x)
```

```
▶ (3) [1, 2, 3]
▶ (2) [2, 3]
```

جهت تغییر مقدار یک index هم کافیست که آن را برابر مقدار جدید قرار دهیم :

```
let array_x = [1, 2, 3]
console.log(array_x)
array_x[0] = "x"
console.log(array_x)
```

```
▶ (3) [1, 2, 3]
▶ (3) ['x', 2, 3]
```

حلقه ها در برنامه نویسی ؟ در زبان‌های برنامه نویسی عباراتی وجود دارند که به آن‌ها حلقه ها یا Loop گفته می‌شود . این دستورات به طور مکرر و به تعدادی که به آن‌ها گفته می‌شود تکرار می‌شود و یک عمل را مکرراً تکرار میکنند . معمولاً دو حلقه اصلی یعنی for و while در زبان‌های برنامه نویسی وجود دارد و در برخی foreach نیز نوعی حلقه است که کاربرد جالبی دارد البته مثل اینکه در جاوااسکریپت حلقه هایی تحت عنوان for…of و for…in و do/while نیز وجود دارد . بریم حلقه while رو بررسی کنیم ببینیم چطوریه .

حلقه while در JavaScript ؟ این حلقه یک شرط را میگیرد و تا زمانی که شرط درست است عبارات درون خود را تکرار میکند و سینتکس آن به شکل زیر است :

```
while(condition){
    Statement_1 ...
    Statement_2 ...
    Statement_3 ...
}
```

تا زمانی که condition درست باشد، Statement_۱, Statement_۲ و Statement_۳ تکرار می‌شوند . مثلاً حلقه زیر برای همیشه تکرار میشود، چرا که true همیشه صحیح است و هیچ‌وقت false نمیشود :

```
while (true){
    console.log("It will repeat for ever, Like a friendship that never dies .")
}
```

و حلقه زیر هیچ‌وقت تکرار نمیشود، چرا که false همیشه false است و چیزی که false است true نمیشود :

```
while (false){
    console.log("It won't repeat at all, Like a baby that never born .")
}
```

یا مثلاً دستور زیر به مقدار length ارایه تعریف شده تکرار می‌شود و هر بار یک index از ارایه را console.log میکند . متغیر i به عنوان شمارنده هر بار یک واحد بیشتر می‌شود تا در نهایت به جایی برسد که شرط داخل while اشتباه شود و حلقه به پایان برسد :

```
let array_1 = [1, 2, 3, 4]

let i = 0
while (i < array_1.length){
    console.log(array_1[i])
    i += 1
}
```

```
1
2
3
4
```

در حلقه ها دستوری وجود دارد تحت عنوان break که زمانی استفاده می‌شود که بخواهیم حلقه را در شرایطی خاص قطع کنیم . مثلاً فرض کنید که یک ارایه شامل اعداد ۰ تا ۱۰ را داریم . یک حلقه ایجاد میکنیم و میگوییم که همه آن‌ها را console.log کند . اما یک شرط قرار میدهیم که اگر عدد مورد نظر ما ۴ شد، کلاً حلقه را بشکن و ازش خارج شو . این دستور بسیار استفاده می‌شود و بهتر است که مفهوم آن را درک کنیم :

```
let array_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

let i = 0
while (i < array_1.length){
    if (array_1[i] == 4){
        break
    }
    console.log(array_1[i])
    i += 1
}
```

```
1
2
3
```

می‌بینید که وقتی array[i] به عدد چهار میرسد شرط ما درست می‌شود و محتویات آن اجرا می‌شود که دستور break است و دیگر به console.log هم نمیرود و از حلقه خارج می‌شود .
دستور دیگری که در حلقه ها استفاده می‌شود continue است و این دستور یعنی اینکه اوکی، از این مورد بگذر و برو سروقت مقدار بعدی، همانطور که در مثال زیر می بینید، گفتیم اگر array_۱[i] شد ۲، به مقدار I یک واحد اضافه کن و برو سروقت مقدار بعدی و نمیخواد که console.log کنی :

```
let array_1 = [1, 2, 3, 4, 5]

let i = 0
while (i < array_1.length){
    if (array_1[i] == 2){
        i += 1
        continue
    }
    console.log(array_1[i])
    i += 1
}
```

```
1
3
4
5
```

و همانطور که میبینید وقتی ۲ رسیده است آن را console.log نکرده است و رفته است سروقت ۳ . از این دستور نیز زیاد در حلقه ها استفاده می‌شود . این دو دستور کمی مفهومی هستند ولی بسیار کارا و مفید . بهتر است همینجا یاد بگیرید .

[۰۱:۴۶:۲۵]

حلقه for در جاوااسکریپت ؟ حلقه for نسبت به حلقه while تفاوتهایی دارد . در حلقه while ما از تعداد تکرار آگاهی نداشتیم و فقط یک شرط رو قرار میدادیم و تکرار انجام میشد . شمارنده رو در حلقه while باید قبل از حلقه تعریف میکردیم و شاید هم هیچ‌گاه تعریف نکنیم . در حلقه for که سینتکسی به شکل زیر دارد، شمارنده ابتدا تعریف میشود، سپس شرط نوشته می‌شود و گامهای شمارنده هم تعیین خواهد شد :

```
for (initialExpression; condition; updateExpression) {
    Statement_1...
    Statement_2...
    Statement_3...
    ...
}
```

initialExpression همان تعریف کردن شمارنده است که مثلاً میگوییم let j = ۰، شرط تکرار حلقه condition است که مثلاً میگوییم j < ۵ و در قسمت updateExpression گامهایی که شمارنده طی میکند را تعیین میکنیم، مثلاً میگوییم j++ که یعنی هر بار یک واحد به j اضافه کند یا j += ۲ که یعنی هر دور دو واحد به j اضافه شود .

```
for (let j = 0; j < 5; j++){
    console.log(j)
for (let j = 0; j < 5; j += 2){
    console.log(j)
}
0
2
4
```

میتوانیم از حلقه for جهت دسترسی به index های یک ارایه نیز استفاده کنیم . در مثال زیر این کار رو انجام داده‌ایم و این کار بسیار رایج است :

```
let array_1 = [1, 2, 3, 4, 5]

for (let i = 0; i < array_1.length; i++){
    console.log(array_1[i])
}
1
2
3
4
5
```

در حلقه for نیز به مانند حلقه while و دیگر حلقه ها میتوانیم از continue و break استفاده کنیم .

حلقه for...of در جاوااسکریپت ؟ این حلقه به صورت اختصاصی برای کار بر روی ارایه های و index های آن‌ها طراحی شده است . در زبان‌های برنامه نویسی چون php که در آینده میبینیم ما حلقه ای تحت عنوان foreach داریم که این کار را انجام میدهد . در جاوااسکریپت foreach نیز وجود دارد که به عنوان یک متد بر روی ارایه هاست و استفاده از آن نیاز به کمی دانش برنامه نویسی جاوااسکریپت دارد ولی به صورت ساده میتوانیم for...of رو استفاده کنیم و سینتکس آن به شکل زیر است :

```
let array_1 = [value1, value2, value3, value4, value5]

for (const val of array_1){
    console.log(val)
}
```

مثال بالا یک ارایه به نام array_۱ با ۵ مقدار تعریف کرده است . حلقه for...of به معنی : به ازای هر مقدار داخل array_۱ که نامش را val میگذاریم اجرا می‌شود . یعنی ابتدا val می‌شود value۱ و بعد val می‌شود value۲ تا در نهایت val می‌شود value۵ و بر روی آن console.log می‌شود و حلقه به پایان میرسد .

```
let array_1 = [1, 2, 3, 4, 5]

for (const val of array_1){
    console.log(val)
}
```

```
1
2
3
4
5
```

توابع یا function ها در جاوااسکریپت ؟ توابع از جالب توجه ترین چیز هایی هستند که در زبان‌های برنامه نویسی وجود دارند و موجب می‌شوند که کدهای برنامه نویس مرتب تر و خوانا تر شود . فرض کنید که در طول کد شما قرار است ۳ کار مختلف انجام شود . شما میتوانیم کدهای این ۳ کار مختلف رو پشت سر هم بنویسید تا در نهایت به انتها برسد و در یک فایل ذخیره کنید و استفاده نمایید . مثلاً به شکل تصویر زیر عمل کنیم :

```
//Job 1 Statements
Job_1
Job_1_1
Job_1_1_1
Job_1_1_2
Job_1_2
Job_1_2_1
Job_1_3
Job_1_4
// Job 2 Statements
Job_2
Job_2_1
Job_2_1_1
Job_2_1_2
Job_2_2
Job_2_2_1
Job_2_3
Job_2_4
// Job 3 Statements
Job_3
Job_3_1
Job_3_1_1
Job_3_1_2
Job_3_2
Job_3_2_1
Job_3_3
Job_3_4
```

حال فرض را بر این بگیریم که میخواهیم Job ۲ رو بعد از Job ۳ دوباره تکرار کنیم و خب باید بیاییم و Statement های آن را دوباره تکرارکنیم :

```
//Job 1 Statements
Job_1
Job_1_1
Job_1_1_1
Job_1_1_2
Job_1_2
Job_1_2_1
Job_1_3
Job_1_4
// Job 2 Statements
Job_2
Job_2_1
Job_2_1_1
Job_2_1_2
Job_2_2
Job_2_2_1
Job_2_3
Job_2_4
// Job 3 Statements
Job_3
Job_3_1
Job_3_1_1
Job_3_1_2
Job_3_2
Job_3_2_1
Job_3_3
Job_3_4
// Job 2 Again Statements
Job_2
Job_2_1
Job_2_1_1
Job_2_1_2
Job_2_2
Job_2_2_1
Job_2_3
Job_2_4
```

هر بار که میخواهیم یک کار را تکرار کنیم باید کدهای آن را تکرارا نویسیم . این کار میتواند موجب از بین رفتن خوانایی کد، وجود حفرات امنیتی بیشتر، سنگینی اجرای کد و ... شود که برای یک وبسایت اصلاً چیزهای جالبی نیستند . مثلاً ممکن است در انتها نگاه کنیم و ببینیم که سه چهار بار یک قطعه کد را تکرار کرده‌ایم و موجب شده است که حجم فایل js ما زیاد شود و تعداد خطوط برنامه ما چندین برابر . برای جلوگیری از این مشکل توابع ایجاد شدند . توابع قطعه کد هایی اند که نوشته می‌شوند و هربار که نیاز باشد استفاده می‌شوند . برای دوباره استفاده کردن نیازی به نوشته شدن دوباره ندارند و کافیست که نامی که به آن تابع میدهیم را صدا بزنیم . برای تعریف یک تابع در جاوااسکریپت روش‌هایی وجود دارد که ساده‌ترین روش استفاده از کلمه کلیدی function است . سینتکس کلی تعریف تابع به شکل زیر است :

```
function function_name(arg1, arg2, arg3, ...){
    // Do Something
    // ...
}
```

مثلاً در مثال زیر یک تابع تعریف کردیم به نام msg که هر بار که صدایش میزنیم برای ما عبارت Hello World را console.log میکند :

```
function msg(){
    console.log("Hello World")
}
```

یا هم در مثال زیر ما یک تابع تعریف کردیم به نام sum که دو ورودی میگیرد و سپس مجموع آن دو را console.log میکند :

```
function sum(a, b){
    console.log(a + b)
}
```

در پرانتز روبروی نام تابع ورودی هایی که ممکن است تابع برای عمل کردن نیاز داشته باشد را مینویسیم و میتوانیم آن‌ها را از کاربر بگیریم و در تابع خود اعمال کنیم . حال که با تعریف کردن تابع آشنا شدیم باید صدا زدن آن را یادبگیریم، تابعی که تعریف شود ولی هیچ‌گاه صدا زده نشود که فایده‌ای ندارد . برای صدا زدن تابع کافیست که نام تابع را بنویسیم و اگر ورودی دارد در پرانتز روبروی آن وارد کنیم :

```
function_name(arg1=value1, arg2=value2, arg3=value3, ...)
```

میتوانید مقادیری ورودی تابع را بدون نوشتن نام ورودی یعنی arg۱, arg۲, arg۳ , ... نیز وارد کنید و کافیست که به ترتیب آن‌ها را به تابع بدهیم :

```
function_name(value1, value2, value3, ...)
```

حال اگر بخواهیم تابع msg را که کمی بالاتر تعریف کردیم صدا بزنیم می‌توانیم به شکل زیر عمل کنیم، چون هیچ ورودی ندارد هیچ چیزی به آن نمیدهیم :

```
msg()
```

```
Hello World
```

و اگر بخواهیم تابع sum را صدا بزنیم کافیست که به شکل زیر عمل کنیم و چون دو ورودی دارد بایستی آن‌ها را نیز وارد کنیم :

```
sum(a=12, b=14)
```

```
26
```

```
sum(12, 14)
```

```
26
```

یک کلمه کلیدی به نام return در توابع وجود دارد که بسیار استفاده می‌شوند . این دستور زمانی استفاده می‌شود که شما میخواهیم از تابع به صورت کامل خارج شوید و از یه جایی به بعد دیگر ادامه پیدا نکند . در مثال زیر گفته‌ایم که یک ورودی از کاربر بگیر و اگر این ورودی ۵ بود return کن و از تابع خارج شو ولی اگر ۵ نبود آن را console.log کن :

```
function func1(number) {
    if (number == 5){
        return
    }
    console.log(number)
}

func1(1)
func1(2)
func1(5)
func1(6)
```

```
1
```

```
2
```

```
6
```

میبینید که وقتی ورودی ۵ را وارد میکنیم تابع به دستور return میرسد و کاملاً خارج می‌شود . البته return یک عمل اصلی را انجام میدهد و آن هم برگرداندن یک مقدار از تابع است . فرض کنید که یک تابع دارید که میخواهید بعد از اجرا مقدار نهایی آن را در یک متغیر ذخیره کنید . مثلاً در مثال زیر میبینید که ما میخواهیم مقدار تابع وقتی ورودی ۱۵ و ۲۵ را به آن میدهیم در یک متغیر به نام s ذخیره کنیم و سپس آن را console.log نماییم .

```
function sum(a, b){
    let z = a + b
}

let s = sum(15, 25)
console.log(s)
```

```
undefined
```

میبینید که مقدار s برابر undefined است که یعنی مجموع ۲۵ و ۱۵ در آن ذخیره نشده است . حال چگونه a + b را در s ذخیره کنیم . کافیست که مقدار نهایی را در تابع به return بدهیم تا آن را بازگردانی کند تا بتوانیم آن را در یک متغیر ذخیره کنیم :

```
function sum(a, b){
    let z = a + b
    return z
}

let s = sum(15, 25)
console.log(s)
```

```
40
```

دیدید که وقتی مقدار نهایی را return کردیم توانستیم آن را در یک متغیر ذخیره کنیم . این کار اصلی return است که بسیار کاربردی و مفید خواهد بود . بهتر است درست تمرین کنید تا درست یاد بگیرید .
فرض کنید تابعی را تعریف کرده‌ایم که دو ورودی میگیرد . میخواهیم قبل از انجام هر عملی بر روی ورودی ها بررسی کنیم که آیا ورودی ها وارد شده‌اند یا خیر ؟ برای این کار میتوانیم از یک دستور شرطی در تابع استفاده کنیم . وقتی یک ورودی وارد نشده باشد typeof آن undefined خواهد بود و همینطور مقدار آن معادل false خواهد بود، پس مینویسیم :

```
function func_1(arg1, arg2){
    if(arg1 && arg2){
function func_1(arg1, arg2){
    if(typeof(arg1) !== "undefined" && typeof(arg2) != "undefined"){
        return "We have two arguments ."
    }else {
        return "No arguments found ."
    }
}
```

API چیست ؟ Application Programming Interfaces رابطی است که میتواند مابین یک نرم‌افزار یا نرم افزای دیگر، یک نرم‌افزار با سخت‌افزار باشد که موجب شود آن دو بتوانند اطلاعاتی را با هم به اشتراک بگذارند . فرض کنید یک نرم‌افزار تحت وب داریم که یک وبسایت رمز ارز می‌باشد . این وبسایت قیمت رمز ارزها را به صورت Real Time در خود ارائه میکند . ما اگر بخواهیم از این قیمت ها در نرم‌افزار خود استفاده کنیم به صورتی که منبع ما وبسایت رمز ارز باشد بایستی از API آن وبسایت رمز ارز استفاده کنیم . این API ها توسط برنامه نویس ها برای هر نرم افزاری که میخواهند با آن تعامل داشته باشند نوشته می‌شوند . مثالی دیگر این است که فرض کنید که یک وبسایت داریم و میخواهیم اپلیکیشن موبایلی آن وبسایت را نیز طراحی کنیم . برای طراحی اپلیکیشن موبایلی میتوانیم یا بیاییم به صورت مستقیم به پایگاههای داده مان متصل شویم و بدون استفاده از API وبسایت بنویسیم و یا میتوانیم برای وبسایت یک API طراحی کنیم و از آن API در اپلیکیشن موبایلی بهره بگیریم . API ها زبانی یکسان ما بین دو طرف هستند و موجب می‌شوند که آن دو طرف بتوانند به یکدیگر داده انتقال دهند و از یکدیگر داده بگیرند .

حال که API رو فهمیدیم موقع آن است که درمورد API های مرورگر ها صحبت کنیم . مرورگرها تعداد خیلی زیادی API دارند که آن‌ها در زبان جاوااسکریپت قابل استفاده اند . یعنی ما میتوانیم از طریق API مرورگرها به وسیله زبان جاوااسکریپت با مرورگرها تعامل داشته باشیم . این API ها جزو هسته اصلی زبان جاوااسکریپت نیستند ولی در لایه‌ای بالاتر از Core زبان جاوااسکریپت وجود دارند . Browser API های هر مرورگر ممکن است تفاوتهایی داشته باشد و آن‌ها را توسط زبان‌های سطح پایین مثل C++ در کروم، Rust در Mozilla Firefox و ... مینویسند . درنهایت بگویم که API ها موجب می‌شوند که پیچیدگی ها بسیار کمتر شوند چرا که راه حلی برای دسترسی به چیزی ارائه میکنند که اگر این راه حل ارائه نشود پیچیدگی بسیار زیادی بوجود میاید . مثلاً اگر بخواهیم بدون استفاده از Browser API صدا انتقال دهیم ممکن است به تعداد خیلی زیادی کد نیاز داشته باشیم و شاید هم اصلاً امکانش وجود نداشته باشد ولی Browser ها API مختص انتقال صدا دارند که میتوانیم از آن‌ها استفاده کنیم تا صدا را انتقال دهیم .

Broswer API ها شامل موارد زیر می‌شوند :

۱. API جهت تغییر Document ها
۲. API هایی که جهت Fetch کردن داده از Server استفاده می‌شوند
۳. API هایی که جهت کشیدن و تغییر دادن Graphic ها استفاده می‌شوند
۴. API های صدا و تصویر
۵. API های مربوط به دیوایس
۶. API های استوریج سمت Client ← مثل Local Storage, Sessions & Cookies, Session Storage in Chrome, WebSQL in Chrome (Depricated but exists)

Asynchronous Programming چیست ؟ زبان برنامه نویسی جاوااسکریپت یک زبان تک thread است، thread ها چیز هایی هستند که قطعه کد های ما را با شروع شدنشان اجرا میکنند، این thread ها کدهای ما را درصورت نیاز به پردازش به CPU میدهند و نتیجه را برمیگردانند، جاوااسکریپت وقتی اجرا می‌شوند تنها یک thread دارد . ما مبحثی داریم به نام Multi Threading که میتوانیم چندین thread داشته باشیم که هرکدام کار خاصی را انجام دهد . Multi Threading در جاوااسکریپت وجود ندارد و قاعدتاً یک thread اجرا می‌شوند و خط به خط کد را تا انتها اجرا میکند و این مشکلی را بوجود میاورد ! اگر ما در کد خود یک پردازش سنگین داشته باشیم که نیازمند زمان بیشتری برای انجام هست برنامه تا زمان پردازش این فرایند سنگین منتظر میماند تا پردازش آن به اتمام برسد و بعد به سراغ کدهای بعدی می‌رود . شاید ما به عنوان برنامه نویس نمیخواهیم که برنامه برای پردازش آن فرایند سنگین منتظر بماند و میخواهیم که بقیه کد را اجرا کند و آن پردازش هم در خط زمانی دیگری به فرایند خود ادامه دهد تا به پایان برسد . برای چنین کاری اصطلاحی تحت عنوان Asynchronous Programming بوجود آمد که از لحاظ لغوی به معنی برنامه نویسی غیر متقارن است . setTimeout یکی از مواردی است که در جاوااسکریپت اضافه شده و از API های مرورگرها استفاده میکند و امکان Asynchromous Programming را فراهم میکند . فرض کنید دوتابع داریم که اولین تابع یک پردازش نسبتاً سنگینی را باید انجام دهد و تابع دوم پردازش آنچنانی ندارد و این دوتابع پشت سر هم صدا زده می‌شوند . به صورت عادی تابع دوم می بایست تا اجرا شدن کامل تابع اول صبر کند و سپس اجرا شود .

```
> function func_1(){
      for(let i = 0; i < 10000; i++){
          console.log("HI")
      }
  }

  function func_2(){
      console.log("Bye Bye")
  }

  func_1()
  func_2()
```

```
10000 HI
Bye Bye
```

می‌بینم که تابع func_۲ تا زمانی که تابع func_۱ به اتمام نرسید اجرا نشد . این دلالت بر همان یک thread بودن جاوااسکریپت دارد و خب گفتیم میتوانیم از طریق setTimeout این مشکل را حل کنیم .

```
> function func_1(){
      setTimeout(() => {
          for(let i = 0; i < 10000; i++){
              console.log("HI")
          }
      }, 1000)

  }

  function func_2(){
      console.log("Bye Bye")
  }

  func_1()
  func_2()
```

```
Bye Bye
undefined
10000 HI
```

حتی اگر تابع دوم هم دارای پردازش زیادی باشد به صورت غیر متقارن اجرا میشوند، تابع اول یک ثانیه صبر میکند و در این یک ثانیه تابع دوم شروع کرده است به پردازش و بعد از یک ثانیه در کنار تابع دوم تابع اول نیز شروع میکند به پردازش و همزمان هر دو در حال پردازش هستند :

```
> function func_1(){
      setTimeout(() => {
          for(let i = 0; i < 10000; i++){
              console.log("Hello friend")
          }
      }, 1000)

  }

  function func_2(){
      for(let j = 0; j < 10000; j++){
          console.log("Bye")
      }
  }

  func_1()
  func_2()
```

```
10000 Bye
undefined
10000 Hello friend
```

به عبارتی میگن که وقتی یک تابع میخواهد اجرا شود در محیطی به نام Call Stack قرار میگیرد و توابع قبل از Call Stack در صف یا Queue قرار میگیرند . وقتی موقع اجرای یک تابع میرسد آن‌ها اجرا می‌شوند و وقتی به انتها رسیدند از Call Stack خارج می‌شوند و نوبت به تابع بعدی

میرسد . این خود موجب یک thread بودن جاوااسکریپت است ولی وقتی ما از Browser API هایی مثل setTimeout استفاده میکنیم، تابع در WEB API قرار میگیرد، در آنجا پردازش سنگین خود را انجام میدهد و سپس درQueue قرار میگیرد و صبر میکند تا تابعی که در Call Stack درحال اجراست به پایان برسد و چیزی به نام Event Loop بررسی میکند که آیا Call Stack خالیست یا نه و اگر خالیست تابعی که از Web API به Queue آمده را در Call Stack قرار میدهد و خروجی آن را به ما نمایش خواهد داد .
اگر بخواهم یه طوری دیگه توضیح بدم باید بگم که اگر از Asynchronous Programming استفاده نکنیم، هر تابعی که اجرا می‌شود قبل از اجرا شدن در یک Queue قرار میگیرد و این Queue صفیست از توابعی که باید اجرا شوند و وقتی نوبت به یک تابع رسید برای اجرا شدن به Call Stack فرستاده می‌شود و در آنجا اجرا می‌شود و وقتی اجرایش به پایان رسید خروجی را نشان میدهد و از Call Stack خارج می‌شود و بعد نوبت به تابع بعدی که در Queue است میرسد اما وقتی ما از Asynchronous Porgramming استفاده میکنیم یک محیط دیگر به نام WEB API اضافه می‌شود و یک تابع که قرار است به صورت Asynchronous اجرا شود در آن محیط قرار میگیرد و اجرا میشود، به صورت همزمان با تابعی که در Call Stack در حال اجراست، وقتی تابع داخل WEB API اجرا شد به Queue منتقل می‌شود و در صف قرار میگیرد تا بتواند به Call Stack برود و خروجی خود را نشان دهد، این تابع دیگر در Call Stack اجرا نمیشود و فقط خروجی آن نشان داده می‌شود .

[۰۲:۲۳:۴۷]
Anonymous Function ها یا توابع بینام چیستند ؟ به توابعی گفته می‌شود که نام ندارند، به همین سادگی . این توابع به دو صورت در جاوااسکریپت تعریف می‌شوند :

```
(function(par1, par2, par3) {
    //Statement1
    //Statement2
    //...
})("value1", "value2", "value2")
```

همانطور که میبینید کلمه کلیدی function که هیچ نامی جلویش نیست و در پرانتز های جلوی آن Parameter های تابع رو نوشتیم که هر کدام Value خود را دارد . مابین {} عمل‌کرد تابع را تعریف میکنیم . دقت شود که از کلمه کلیدی function تا {} می بایست در () قرار بگیرد وگرنه Syntax Error خواهیم گرفت .

```
function (par1="value1", par2="value2", par3="value3") {
    console.log(par1, par2, par3)
}
Uncaught SyntaxError: Function statements require a function name
(function (par1="value1", par2="value2", par3="value3") {
    console.log(par1, par2, par3)
})
f (par1="value1", par2="value2", par3="value3") {
    console.log(par1, par2, par3)
}
```

میبینید که اگر بلاک تعریف تابع را در () قرار ندهیم آن را به عنوان یک تابع بی نام درنظر نمیگیرد و میگوید که بایستی برای تابع خود نامی را انتخاب کنید ولی اگر بلاک تعریف تابع را در () قرار دهیم آن را به عنوان یک تابع بی نام در نظر میگیرد . اگر به سینتکس تعریف تابع بینام دقت کنید میبینید که یک جفت () نیز در انتها نوشته شده است :

```
(function() {
    //Statement1
    //Statement2
    //...
})()
```

این جهت این است که بعد از تعریف تابع، آن تابع را صدا بزند و اجرا کند . اگر تابع ما Parameter هایی داشته باشد در این پرانتز ها مقادیر مربوط به هر کدام را وارد میکنیم .

```
(function(par1, par2, par3) {
    //Statement1
    //Statement2
    //...
})("value1", "value2", "value2")
```

خب روش دیگری نیز برای تعریف توابع بینام وجود دارد و سینتکس آن به شکل زیر است :

```
() => {
    //Statement1
    //Statement2
    //..
}
```

حتی میتوانیم اگر بلاک تابع ما فقط شامل یک خط کد باشد آن را خلاصه تر نیز تعریف کنیم، به شکل زیر :

```
() => console.log("Hello Friend .")
```

میبینید که تابع typeof نیز آنها را به عنوان function میشناسد :

```
typeof(() => console.log("Hello Friend ."))
'function'
typeof(() => {})
'function'
```

اگر بخواهیم تابع بینامی را که با دو سینتکس بالا تعریف میکنیم را اجرا کنیم کافیست که () را در انتهای بلاک قرار دهیم تا صدا زده شود .

```
(() => {
    console.log("Hello Friend .")
})()
Hello Friend .
undefined
(() => console.log("Hello Friend ."))()
Hello Friend .
```

دقت شود که قبل از صدا زدن حتماً بلاک تابع تعریف شده را در () قرار دهید تا Syntax Error نگیرید . حال که تعریف کردن Anonymous Function ها رو دیدیم بهتر است درمورد کاربرد آن‌ها نیز آگاهی پیدا کنیم، وگرنه چه کاریه اخه ؟ میایم و همه Function ها رو با نام تعریف میکنیم . توابع Anonymous زمانی استفاده می‌شوند که یک تابع مرتبه بالاتر داشته باشیم و این تابع به عنوان ورودی یک تابع دیگر را بگیرد . مثلاً setTimeout رو دیدیم که در ورودی یک تابع Anonymous رو به عنوان Parameter اول خود گرفت . در JavaScript زیاد پیش میاد که برخی توابع به عنوان ورودی تابع بگیرند . در آینده مثال‌های زیادی خواهیم دید پس از تعریف کردن چنین توابعی نترسید و سعی کنید باهاش اخت پیدا کنید .

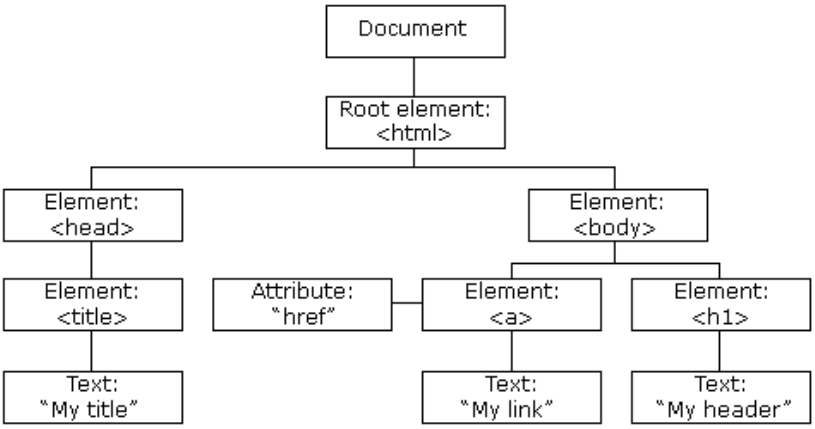
```
setTimeout(() => {
    console.log("Hello Friend .")
}, 1000)
35
Hello Friend .
```

این نکته هم دقت شود که میتوانیم از طریق سینتکس هایی که برای تعریف Anonymous Function ها گفتیم توابع دارای نام را تعریف کنیم .
یعنی درواقع یک تابع بدون نام تعریف کنیم در حالی که به آن نام میدهیم .

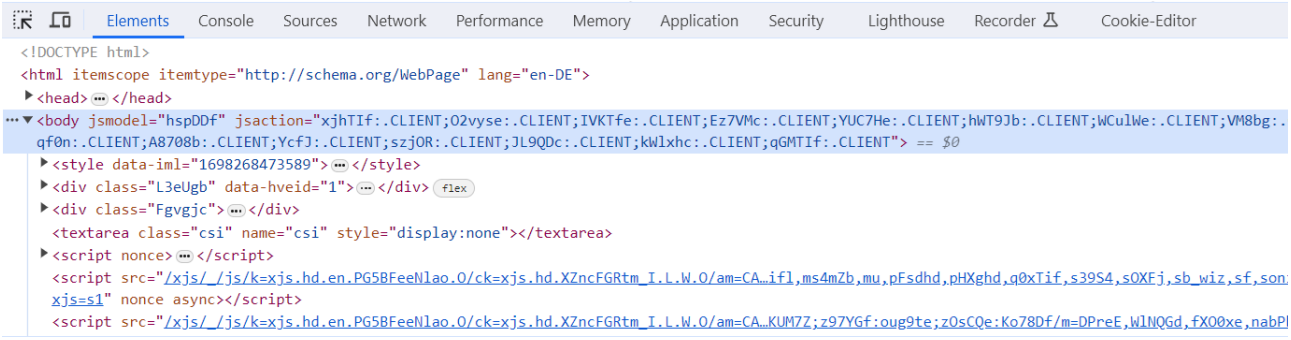
```
const foo = () => {
    //Statement1
const bar = () => console.log("Hello Friend .")
const baz = function () {
    console.log("Hello Friend .")
}
undefined
typeof(baz)
'function'
baz()
Hello Friend .
```

پس ما در جاوااسکریپت چندین روش متفاوت برای تعریف توابع داریم که بهتر است با همه آن‌ها آشنا باشیم چرا که ممکن است در جایی یک برنامه نویس از هر کدام از این‌ها استفاده کرده باشد .

Web API ها و DOM چیست ؟ Web API ها یا WEB Application Programming Interface ها جمعا به Browser API ها و Server API ها میگویند .
مرورگر صفحات وب رو در قالب چیزی به نام DOM یا Document Object Model در خود دارد . DOM یک Data Representation یا نوعی نمایش داده است که در این نوع Object هایی که ساختار و محتویات یک سند (Document) وب را می‌سازند شامل می‌شود .(خودمونی بگم یعنی همون تگهای HTML و محتویات Tag های HTML)



تصویر قبل یک نمونه از چیزی که یک DOM درخود میتواند داشته باشد . میبینید که Tag ها و Attribute تگها و محتویات تگها که Text هستند را در خود دارد . ما به عنوان کسی که میخواهد جاوااسکریپ رو کار کنه باید طریقه کار با DOM رو بلد باشیم . DOM مربوط به هر صفحه رو میتونید با زدن کلید F۱۲ و منوی Elements ببینید که در تصویر زیر نمونه‌ای قرار دادم :



باید بدونیم DOM چطوری در حافظه Memory قرار میگیرد و چگونه میتوان آن را تغییر دهیم یعنی مقداری از آن را تغییر دهیم، چیزی از آن کم کنیم یا چیزی به آن اضافه کنیم . به علت اینکه DOM سند HTML ما را به شکل Node ها و Object ها ارائه میکند، زبان‌های برنامه نویسی مثل JavaScript، ... میتوانند با آن تعامل برقرار کنند . میتوانند اطلاعاتی را از آن بخوانند یا می‌توانند چیز از آن را تغییر دهند . برای تعامل با DOM باید از Web API های موجود استفاده کنیم . در زبان جاوااسکریپت ما این API ها را داریم که لیستی از آن‌ها را در زیر میبینید :

- `document.querySelector()`
- `document.querySelectorAll()`
- `document.createElement()`
- `Element.innerHTML`
- `Element.setAttribute()`
- `Element.getAttribute()`
- `EventTarget.addEventListener()`
- `HTMLElement.style`
- `Node.appendChild()`
- `window.onload`
- `window.scrollTo()`

این دستورات جهت تعامل و ارتباط با DOM هستند و هر برنامه نویس JavaScript می بایست کار با آن‌ها را یاد بگیرد، افرادی چون ما که قصد داریم امنیت کدهای اسناد Web را بررسی کنیم می بایست بیشتر یاد بگیریم، چرا که یک BUG می‌تواند در هر کدام از آن‌ها رخ بدهد . دقت هم شود که کلاً جاوااسکریپت بر طبق DOM کار میکنه و ما برای به کار بردن کد های جاوااسکریپت در صفحه خود از Web API های مربوط به تعامل با DOM زیاد استفاده میکنیم .
برای اینکه DOM رو بهتر و دقیق‌تر یاد بگیرید میتونید به لینک زیر مراجعه کنید :
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

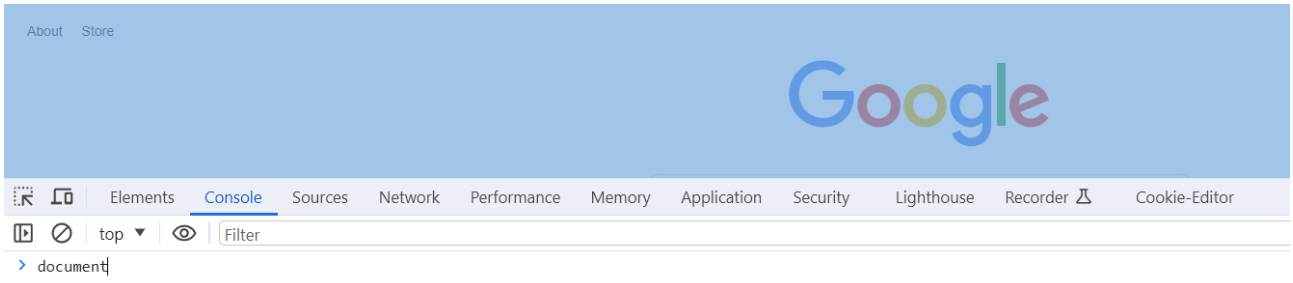
اگر هم بخوام مجموعه‌ای از Web API هایی که میدونم رو بگم میتونم به موارد زیر اشاره کنم :

۱. document.getElementByID
۲. document.getEle*.textContent
۳. document.cookie
۴. document['cookie']
۵. document.domain
۶. document.URL
۷. document.location
۸. window.open
۹. Window.opener
۱۰. localStorage.setItem('x', '۱')
۱۱. localStorage.getItem('x')
۱۲. localStorage.removeItem('x')
۱۳. localStorage.clear()
۱۴. document.write()

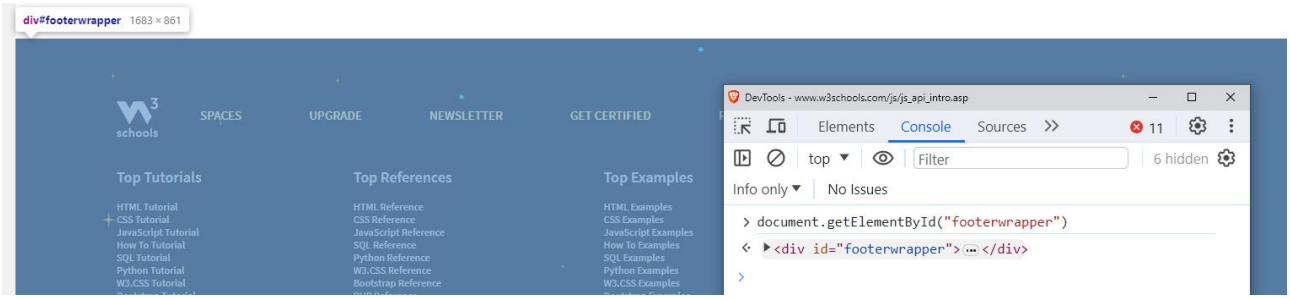
دستور document در جاوااسکریپت ؟ برای اینکه در کدهای جاوااسکریپت خود با DOM بتوانیم تعامل داشته باشیم از این دستور استفاده خواهیم کرد . این دستور مجموعه‌ای از متد ها را به ما میدهد که میتوانیم از طریق آن‌ها با Node ها و Object های DOM ارتباط برقرار کنیم و آن‌ها را تغییر دهیم و یا حتی یک Node را به DOM اضافه کنیم . دستورات زیر بخشی از این متدها هستند :

```
document.getElementById
document.getElementsByClassName
document.getElementsByName
document.getElementsByTagName
document.getElementsByTagNameNS
document.querySelector
document.querySelectorAll
document.body
document.head
document.children
document.childNodes
document.removeChild
document.appendChild
document.createAttribute
document.createElement
document.createTextNode
document.doctype
document.title
document.replaceChild
document.replaceChildren
document.parentElement
document...
```

میتوان گفت که تمام این دستوراتی که در بالا ذکر شد و هرچه هست و ذکر نشده است را میتوان بر روی هر Node دیگری از DOM نیز اجرا کرد . مثلاً متد createElement یک Element جدید در DOM ایجاد میکند و این میتواند بر روی Document یا شد که ریشه و سر DOM است یا بر روی هر Element و تگ دیگری که در جایی از DOM قرار دارد . این دستورات رو زمانی میتوانید بهتر درک کنید که از آن‌ها استفاده کنید . مثلاً سعی کنید که یک صفحه وب رو بدون استفاده از کد های HTML زیاد و فقط از طریق کدهای جاوااسکریپت بنویسید . من این کار رو بارها انجام دادم و اطلاعاتم درمورد DOM بسیار زیادتر شد . یه طورایی حس میکنم انجام این تمرین جذابه، خیلی اوقات واسه تفریح هم که شده این کار رو انجام میدم چون حس میکنم از طریق HTML تگ ایجاد کردن یا تغییر دادن یا حذف کردن کار بسیار ساده‌ای است و درست هم فکر میکنم . گفتیم که document به معنی ریشه اصلی DOM است و فک کنم تا الان توی تصاویر دیدی که DOM ساختاری درختی داره که اولین عضو آن document است و ریشه محصوب میشه اگه DOM رو یک درخت برعکس درنظر بگیرید . شما اگه document رو توی تب Console مرورگرتون بنویسید تمام صفحه یه حالت آبی رنگ میگیره که به معنی انتخاب شدن است و این یعنی document تمام صفحه می‌باشد :



انتخاب کردن یک Object از DOM بر اساس خصیصه id آن در جاوااسکریپت ؟ برای اینکه بتونیم یک Object یا Node از DOM رو تغییر بدیم باید ابتدا اون رو انتخاب کنیم و بعد تغییرات مورد نظر خودمون رو اعمال کنیم . برای انتخاب کردن یک Object از DOM میتوانیم از طریق Attribute های آن انجام دهیم . یکی از این Attribute ها id است و اگر id رو بشناسید میدونید که id برای هر Element در صفحه باید یکتا و منحصر به فرد باشه . اگر بخواهیم یک Element رو بر اساس id انتخاب کنیم باید از متد getElementById استفاده کنیم و به شکل زیر عمل میکنیم :



در کد بالا به document گفتیم که اون Element رو به من بده که id آن برابر footerwrapper است و اون Element همانطور که میبینید رنگش به حالت آبی دراومده و به معنی آینه که انتخاب شده است .

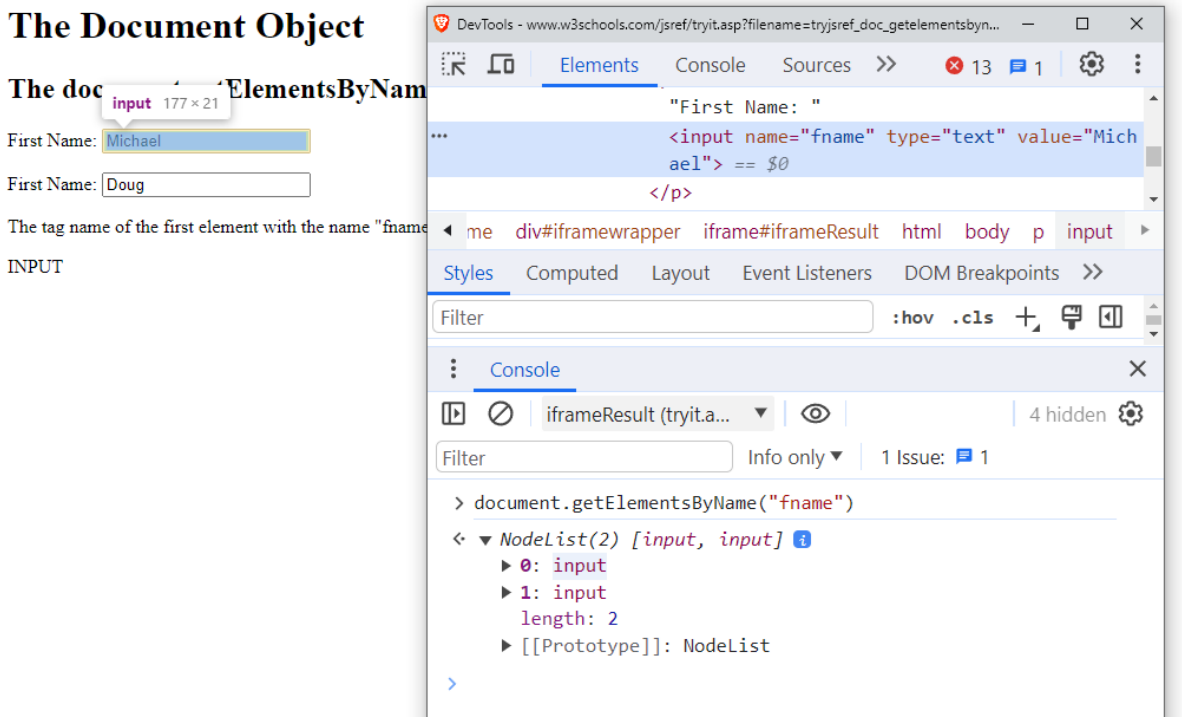
انتخاب Object هایی از DOM از طریق خصیصه class در جاوااسکریپت ؟ یکی دیگر از Attribute هایی که میتوانیم از طریق آن Object های DOM رو انتخاب کنیم class است . اگر class رو میشناسید میدونید که مجموعه‌ای از Element ها میتوانند class یکسانی رو داشته باشند از این رو دستور انتخاب Object ها از طریق class به صورت جمع است . در نمونه قبلی getElementById واژه Element مفرد بود چرا که فقط یک مورد وجود داشت ولی در این نمونه getElementsByClassName واژه Element جمع است، یعنی تعداد میتواند بیشتر از یکی باشد . از این رو بر خلاف متد قبلی که یک Element رو برای ما بر میگردوند، این متد یک مجموعه از Element ها رو برای ما برمیگردونه .

```
document.getElementsByClassName("nextprev")
▼ HTMLCollection(2) [div.w3-clear.nextprev, div.w3-clear.nextprev.w3-center]
  ▶ 0: div.w3-clear.nextprev
  ▶ 1: div.w3-clear.nextprev.w3-center
    length: 2
  ▶ [[Prototype]]: HTMLCollection
```

می‌بینید که نتیجه یک HTMLCollection شامل دو Element است و میتوانیم از همانطور که در ارایه ها به index ها دسترسی می یافتیم در اینجا نیز دسترسی پیدا کنیم، مثلاً :

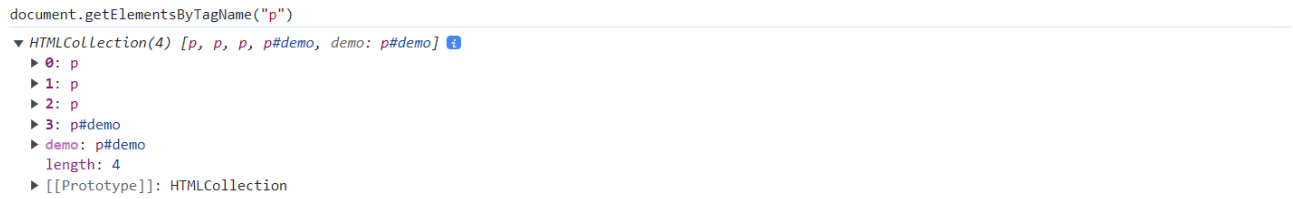
```
let elements = document.getElementsByClassName("nextprev")
console.log(elements[0])
▶ <div class="w3-clear nextprev">…</div>
```

انتخاب Element ها در DOM با خصیصه name ؟ خصیصه name میتواند بر روی مجموعه‌ای از Element های صفحه ما باشد . برای گرفتن مجموعه Element هایی با خصیصه name با مقداری خاص میتوانیم از دستور زیر استفاده کنیم :

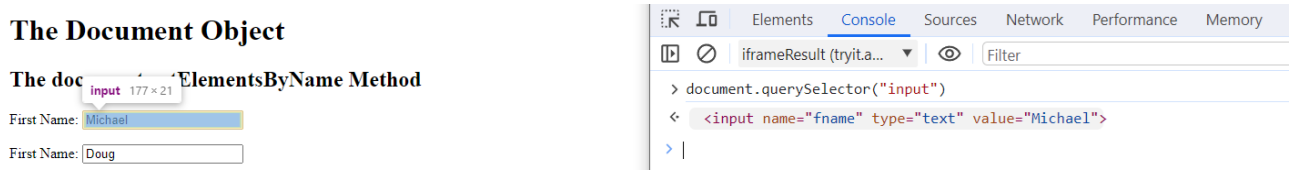


در صفحه وب بالا دو input وجود دارد که name آن‌ها برابر fname است و از document خواستیم که getElementsByName کند آن‌هایی که name آن‌ها fname است و آن دو را انتخاب کرد و برای ما نوشت . ساده است نه ؟

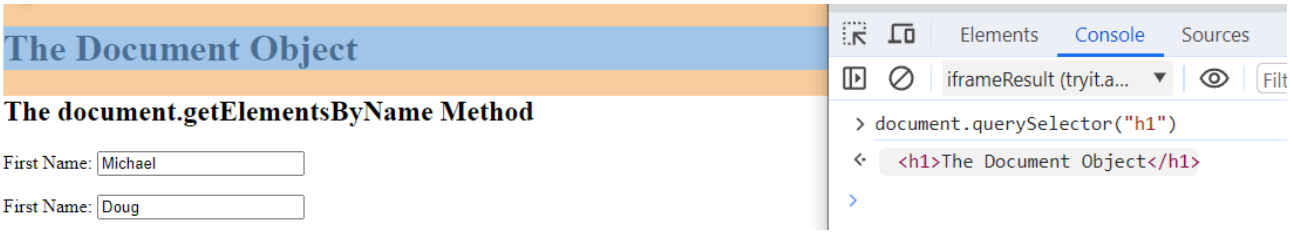
انتخاب مجموعه‌ای از Element ها در DOM که Tag Name یکسانی دارند ؟ فرض کنید میخواهیم تمام Element هایی که p هستند رو انتخاب کنید و بر روی آن‌ها تغییراتی رو اعمال نمایید . برای این کار از document میخواهیم که getElementsByTagName کند p ها رو و در نتیجه برای ما مجموعه‌ای از تگهای span داخل document رو بر میگرداند .



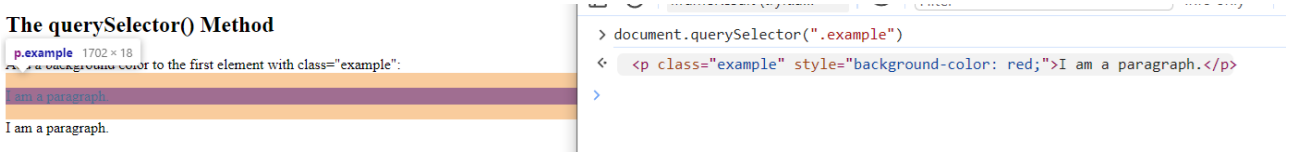
متد querySelector چیست ؟ تا اینجا که درمورد getElement* ها صحبت کردیم بهتر است نسبت به querySelector هم آگاهی داشته باشیم . این متد هم به مانند قبلیا توسط document و هر Element دیگری در صفحه میتواند صدا زده شود و به کار برود، مثلاً میگوییم document.querySelector('QUERY') . این متد طبق QUERY که به آن میدهیم برای ما یک Element رو انتخاب میکند .



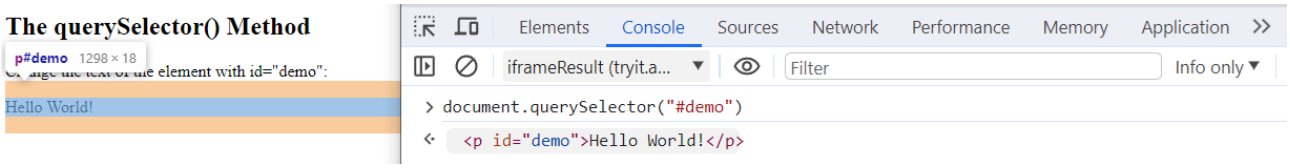
در مثال بالا گفته‌ایم که، آقای querySelector لطفاً اولین Element که اسمش input هست رو انتخاب کن و برگردون و اون هم همین کار رو کرده . پس اگر ورودی querySelector یک مقدار ساده باشد آن را به عنوان نام تگ در نظر میگیرد . مثلاً در نمونه زیر گفتیم که اولین h۱ که می‌بینی رو انتخاب کن و کرده :



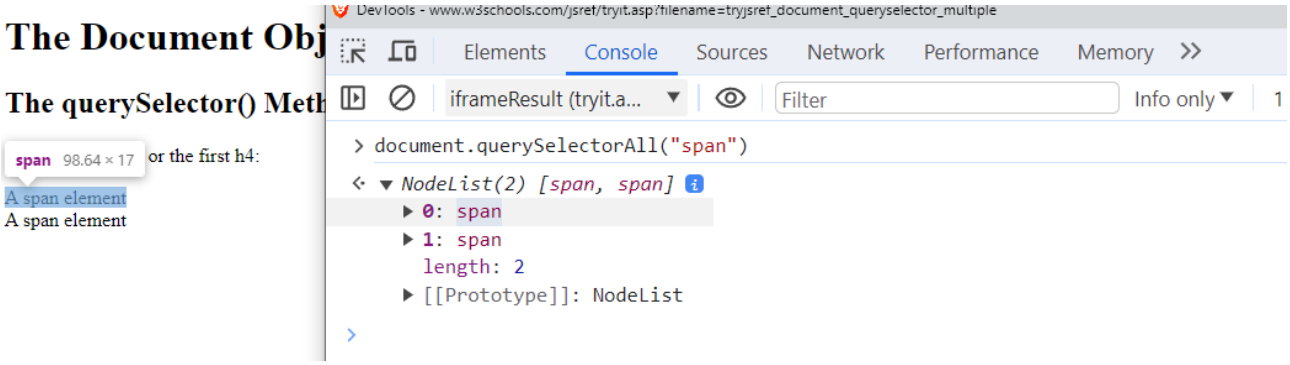
اگر از querySelector بخوایم که مثلاً اولین Element که مقدار خصیصه class آن example است رو به ما بده کافیست که به شکل زیر آن را بنویسیم (.example یعنی کلاسش example باشه) :



اگر یه وقتی خواستیم که اون Element که خصیصه id خاصی رو داره انتخاب کنیم کافیست که مقدار querySelector به شکل زیر باشه (#demo یعنی id طرف demo باشه) :

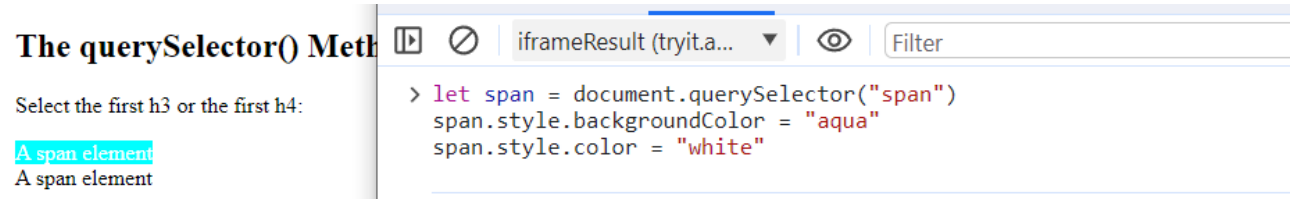


در کنار querySelector یک متد به نام querySelectorAll هم داریم که مجموعه‌ای از Element ها رو بر میگیردونه و تنها تفاوتش با querySelector همینه که querySelector اولین Element مطابق چیزی که بهش دادیم رو بر میگردونه ولی querySelectorAll همه Element های مطابق چیزی که بهش میدیم رو بر میگردونه .

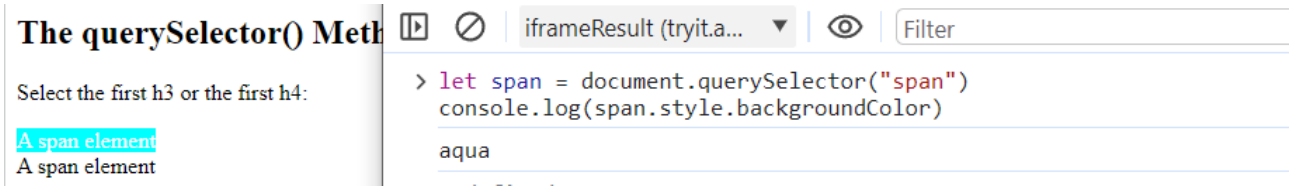


ببینید، selector که میتونید به querySelector یا querySelectorAll بدید بسیار دامنه گسترده ای داره و شما تقریباً میتونید از طریق این متدها هر Object یا Object هایی از DOM رو انتخاب کنید و بر روی آن‌ها تغییرات ایجاد کنید . من همه selector ها رو نمیدونم و حتی نمیتونم همه اونهایی که بلدم رو اینجا توضیح بدم، شما کافیست سرچ کنید و selector مورد نظر خودتون رو پیدا کنید و استفاده کنید، حتی نیازی نیست حفظ کنید چون گوگل همیشه هست .

خب حالا ما یاد گرفتیم که Element ها رو انتخاب کنیم ! حالا چیکار میتونیم با اونها انجام بدیم ؟ تقریباً (میگم تقریباً چون صددرصد مطمئن نیستم) هر کاری که شما میتونید در کد های HTML بر روی صفحاتتون اعمال کنید رو میتونید از طریق JavaScript نیز انجام بدید . مثلاً میتونید Attribute های یک تگ رو تغییر بدید، به اون تگ CSS Style بدید، اون تگ رو حذف کنید، به اون تگ تگهای childeren اضافه کنید و هر چیزی دیگری . مثلاً در نمونه زیر من اولین تگ span رو گرفتم و backgroundColor اون رو به رنگ aqua تغییر دادم و رنگ متن اون رو white کردم :



یا مثلاً گفتم رنگ background تگ span رو console.log کن :



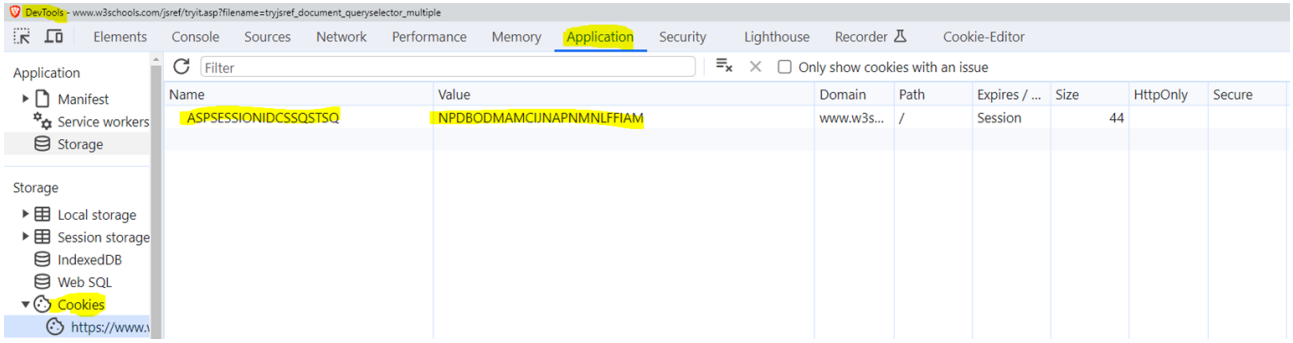
از این کارای قرتی بازی زیاد میشه انجام داد . شما کافیه سعی کنی و یاد بگیری . گفتم که میتونید یک صفحه HTML رو کلاً با کدهای جاوااسکریپت بنویسید و این یعنی همه چیز .

حالا توی Web Application Penetration Testing چه کاربردی داره ؟ یک نمونه مثلاً این هست که شما یک باگ XSS پیدا میکنی، از طریق اون و querySelector مقدار CSRF-Token رو استخراج میکنی و یک حمله CSRF میزنی . هرچی بیشتر جاوااسکریپت یاد بگیرید بهت میتونید Bug پیدا کنید و قاعدتاً واسه کسانی پول براشون اهمیت زیادی داره (فک نکنم کسی باشه که زیاد براش پول مهم نباشه) اگر بیشتر جاوااسکریپت رو بدونی میتونی پول بیشتری بدست بیاری و خب چی بهتر از این .

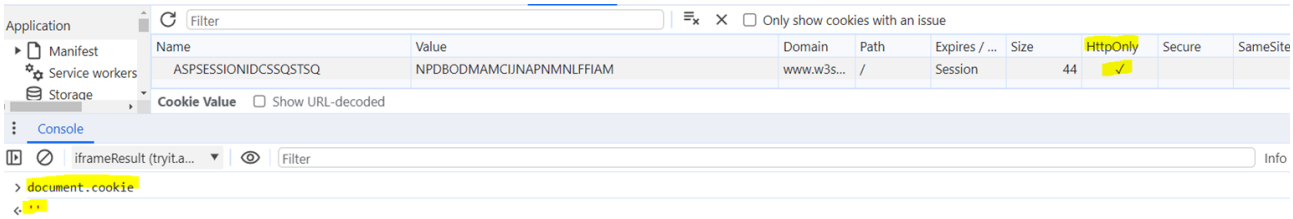
document.cookie چه می‌کند ؟ از طریق این Web API میتونیم به کوکی های اون صفحه دسترسی پیدا کنیم و key و value آن‌ها را بخوانیم :

```
document.cookie
'ASPSESSIONIDCSSQSTSQ=NPDBODMAMCIJNAPNMNLFFIAM'
```

درواقع از طریق این دستور به مقادیر زیر دسترسی پیدا میکنیم :



مثلاً کاربردی که برای این توی حملات میتونم ذکر کنم این هست که میتونیم از طریق این دستور Session ID یک کاربر در یک وبسایت رو Hijack کنیم و به جای اون در اون سایت Login نماییم، بدون Credential اون کاربر و فقط از طریق Session ID . این حمله میتونه از طریق باگ XSS رخ بده و بهش Session Hijacking میگویند و کاری که برای جلوگیری از این حمله میکنن آینه که کوکی ها رو HttpOnly میکنند و دیگه امکان دیدن اونها از طریق document.cookie وجود نداره :



یکی از ویژگی‌های جاوااسکریپت این هست که تقریباً هر چیزی برگرفته از Object هست . یعنی هر چیزی یک Instance از کلاسی به نام Object است .

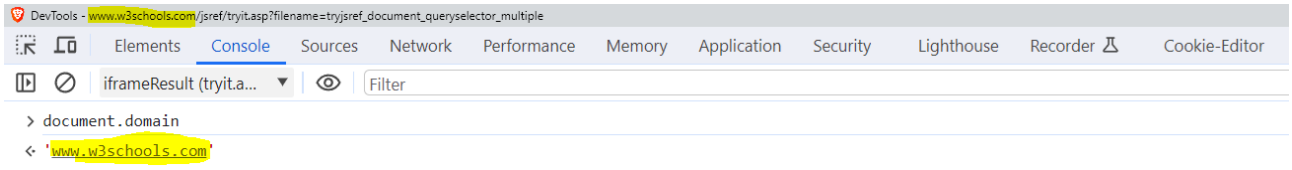
```
typeof(document)
'object'
typeof(document.getElementsByTagName("p")[0])
'object'
typeof(window)
'object'
```

خب این ویژگی چه خوبی داره ؟ گاهی اوقات ممکنه که برنامه نویس یک وبسایت جهت جلوگیری از XSS و مثلاً Session Hijacking بیاد و "." رو بلاک کنه و اجازه نده که شما این کاراکتر رو وارد کنید . چون همه چیز از Object است پس همه چیز ویژگی‌های یک Object رو داره که یکی از ویژگی‌هایی که برای Bypass کردن بلاک بودن "." استفاده میشه این هست که بجای استفاده از این کاراکتر از [] جهت دسترسی به متدها و فیلدها استفاده کنیم .

```
document.cookie
'ASPSESSIONIDCSSQSTSQ=NPDBODMAMCIJNAPNMNLEETAM'
document.getElementsByTagName("p")
▶ HTMLCollection [p]
document['getElementsByTagName']("p")
▶ HTMLCollection [p]
```

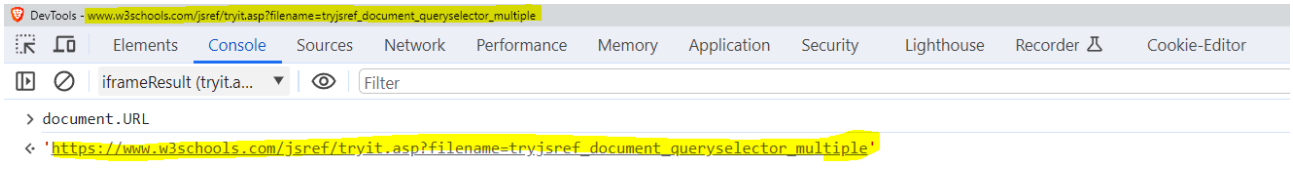
البته این نکته رو هم بگم که اینکه همه چیز از Object ارث بری میکند میتواند در بسیاری از موارد موجب باگ Prototype Pollution شود . در آینده این باگ رو بررسی خواهیم کرد و باگ نسبتاً جالبیه .

document.domain به ما چه میدهد ؟ این Web API به ما Domain Name که در آن قرار داریم را میدهد . مثلاً مثال زیر :



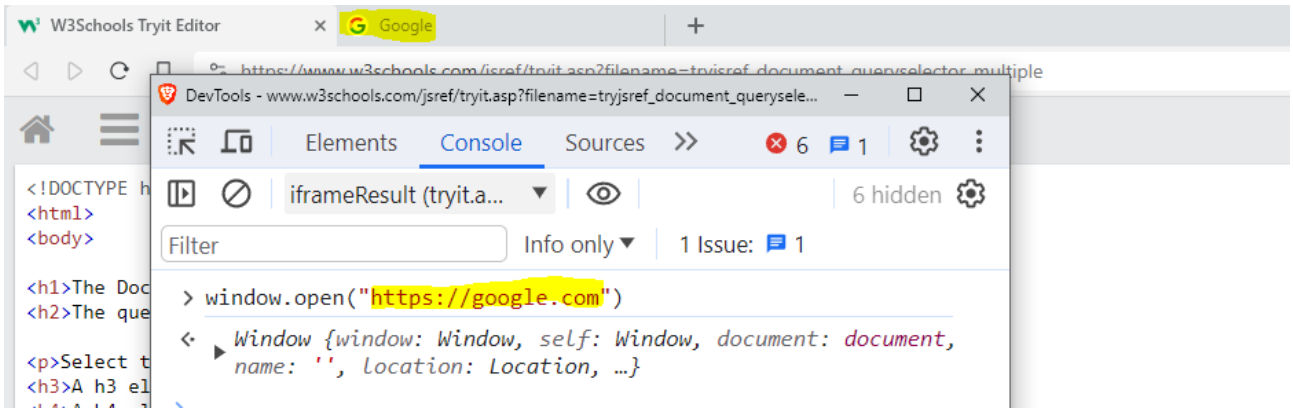
در برخی اوقات ممکن است در یک باگ XSS نیاز شود که بررسی کنیم Domian حاوی باگ چیست و از این دستور میتوانیم بهره بگیریم . اینکه Domain رو بدونیم واسه زمانی که Bug Bounty میزنیم به عنوان POC یا Proof Of Concept برای اثبات آینه که ما مثلاً روی Domain شما این باگ رو زدیم لازمه . در برخی موارد هم واسه تعیین ناحیه ای که آسیب‌پذیر هست میتونیم از این دستور استفاده کنیم .

document.URL چه مقداری را به ما میدهد ؟ این دستور URL کامل، شامل Protocol, Domain, Path, Parameter رو به ما بر میگردونه :

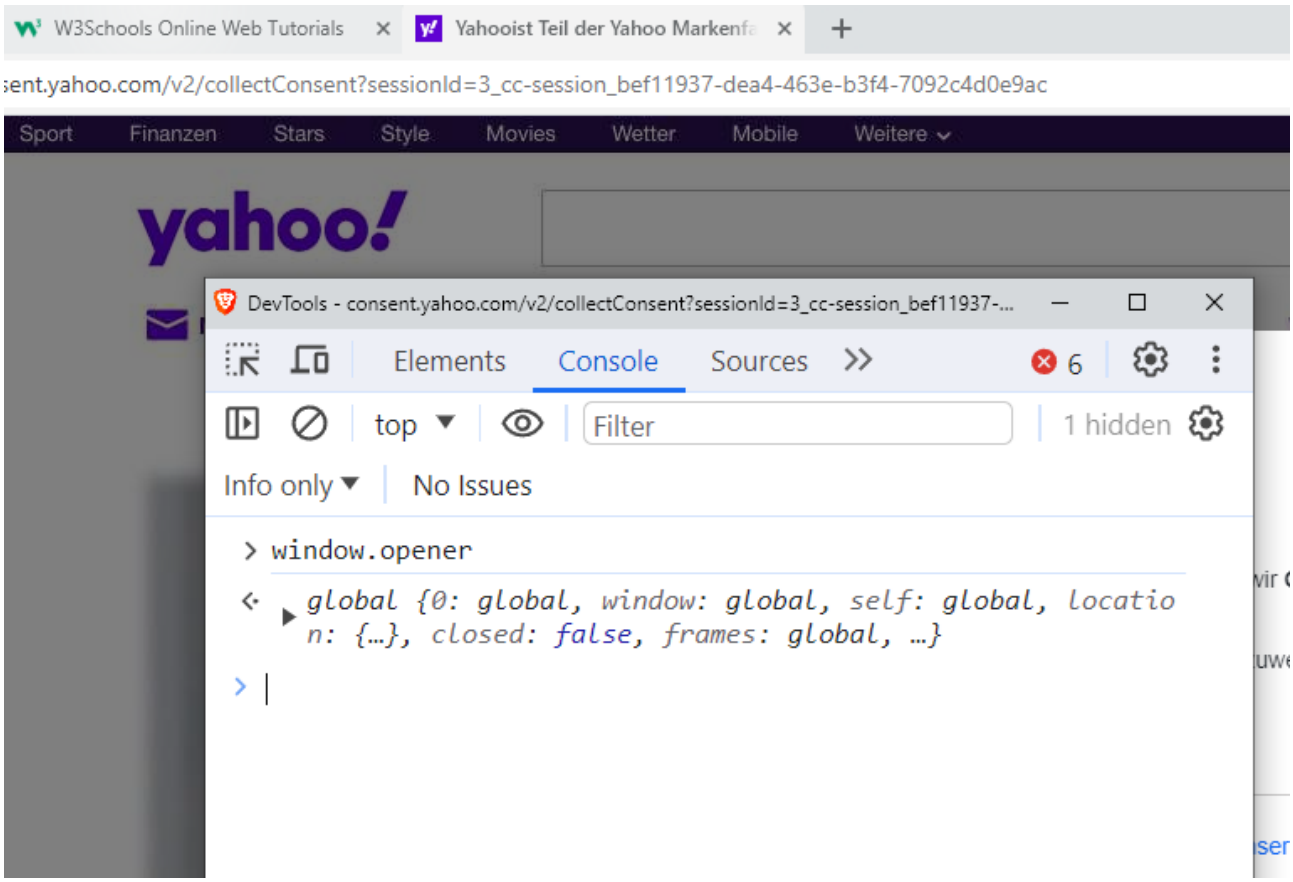


document.location به ما چه میدهد ؟ این دستور یک Object به ما میدهد و URL رو تقسیم میکنه به تکه‌هایی و نام هایی رو روی اونا قرار میده، مثلاً origin چیه ؟ protocol چیه ؟ host چیه ؟ و ...

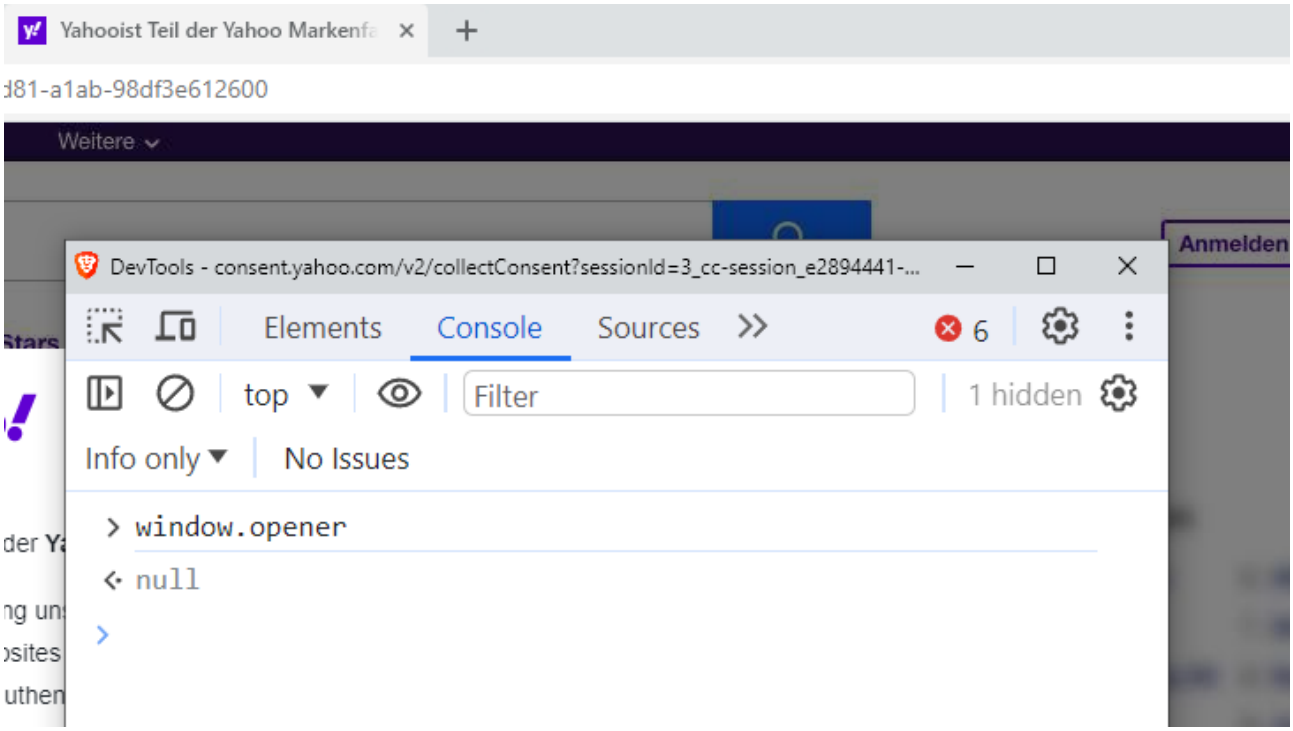
window.open چه میکند ؟ این دستور یک url را میگیرد و آن را در یک tab جدا باز میکند . واسه اکسپلویت کردن Post Message ها استفاده می‌شود که در آینده میبینیم :



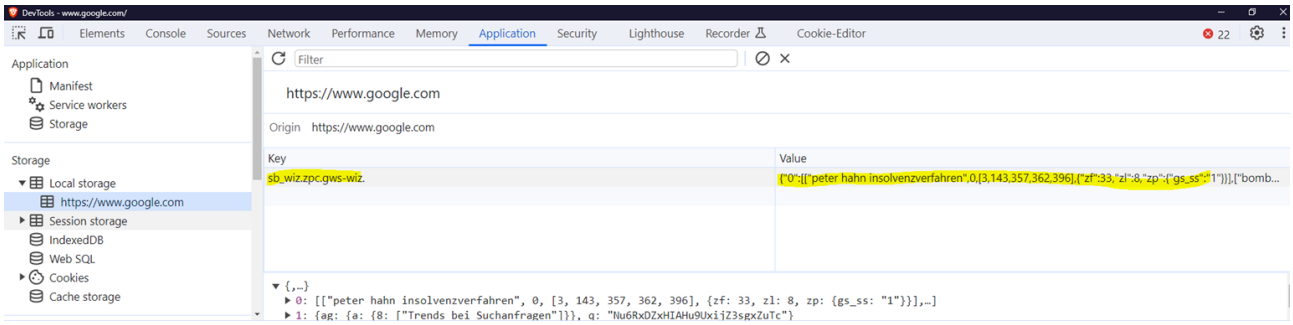
window.opener چه میکند ؟ زمانی که از طریق window.open از یک tab یک tab جدید را باز میکنیم این دستور به ما اطلاعات tab بازکننده tab جدید را میدهد . این دستور در tab جدید باید اجرا شود . مثلاً از w۳schools صفحه yahoo.com رو از طریق window.open بازکردیم . حال به tab یاهو میریم و اونجا توی کنسول میزنیم window.opener :



توی این اطلاعاتی از آن صفحه‌ای است که این صفحه رو باز کرده و اگر یک کاربر به صورت عادی یک tab باز کند و وارد یک سایت شود window.opener مقدار null خواهد داشت :



localStorage چیست ؟ localStorage از Web Api های مربوط به Storage است که مرورگر در اختیار برنامه نویس ها قرار میدهد تا اطلاعاتی بدون حساسیت رو در آن ذخیره کنند . دقت شود که نباید اطلاعات حساس در LocalStorage ذخیره شود چرا که بر خلاف Cookie ها هیچگونه مکانیزم امنیتی بر روی آن اعمال نمیشود . یک هکر میتواند به راحتی با یک باگ XSS بر روی این دادهها تغییراتی را اعمال کند . این دادهها فقط صرفاً یک Key و Value دارد، بدون هیچگونه مکانیزم امنیتی :



localStorage.setItem چه میکند ؟ این متد دو ورودی را میگیرد که اولین نام Key و دومی Value است و در LocalStorage یک Key با آن Value میسازد . مثلاً در مثال زیر ما x با مقدار ۶۹ رو در localStorage ساختیم :

```
localStorage.setItem("x", "69")
```

localStorage.getItem چه میکند ؟ این متد یک Key را به عنوان ورودی میگیرد و سپس مقداری که در آن Key وجود دارد را باز میگرداند :

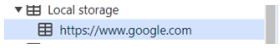
```
localStorage.getItem("x")
'69'
```

localStorage.removeItem چه میکند ؟ این متد یک Key را میگیرد و در صورت وجود داشتن در LocalStorage آن را حذف میکند :

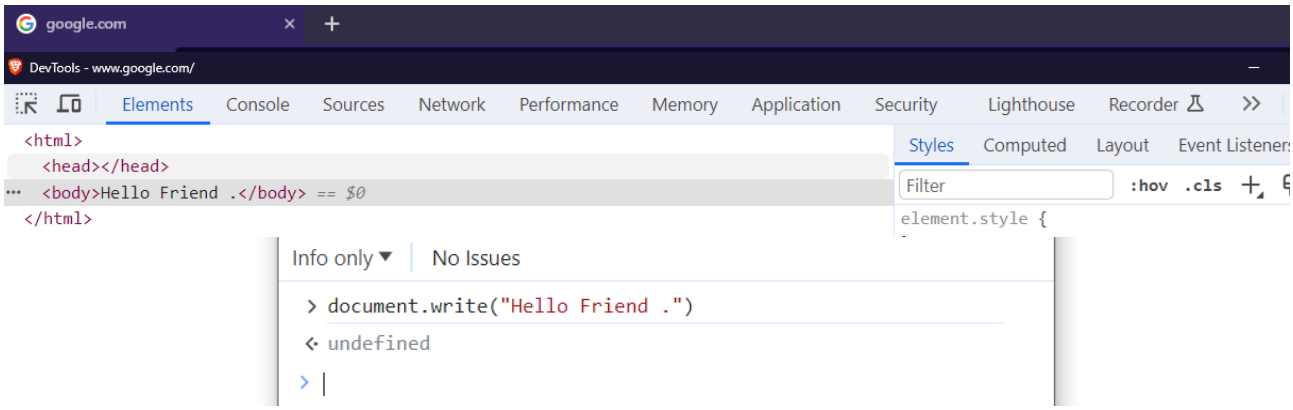
```
localStorage.getItem("x")
'69'
localStorage.removeItem("x")
undefined
localStorage.getItem("x")
null
```

localStorage.clear چه میکند ؟ این متد همه مقادیر ذخیره شده در localStorage را از دم پاک میکند :

```
localStorage.clear()
undefined
```



document.write چه میکند ؟ این متد تمام DOM رو پاک میکنه و یک سند HTML شامل تگهای html, head, body ایجاد میکنه و یک ورودی میگیره که اون رو به عنوان محتویات تگ body قرار میده :



خب فعلاً جاوااسکریپت تا همینجا کافیه و چیزای دیگه رو در آینده درموردش خواهم نوشت . از اینجا به بعد میریم سروقت MySQL چون از دیگر الزامات واسه Web Application Penetration Testing است .

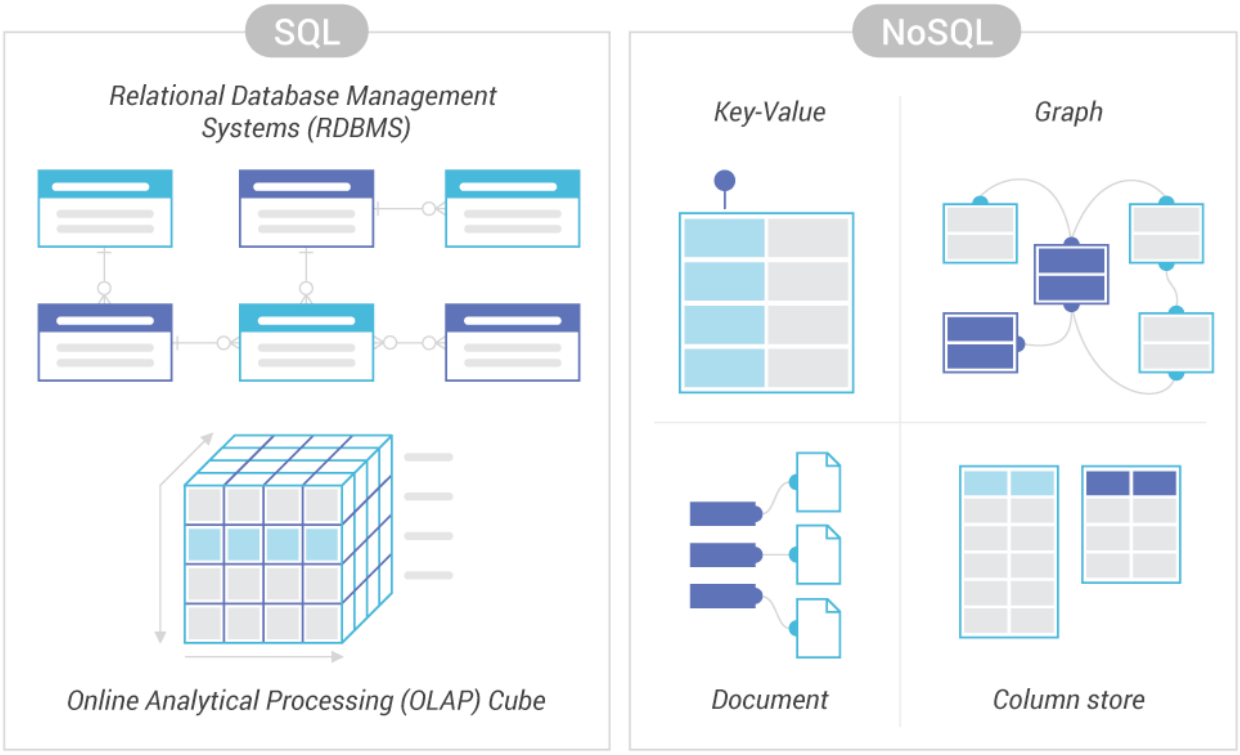
Database - MYSQL
[۰۳:۰۲:۱۲]

Database چیست ؟ مجموعه‌ای سازمان یافته و دارای قواعد از دادهها که میتوان به صورت الکتریکی از طریق سیستم‌های کامپیوتری آن‌ها را ذخیره کرد و یا به آن‌ها دسترسی یافت . جایی که دیتابیس ها پیچیده‌تر می‌شوند معمولاً آن‌ها را از طریق Modeling Technique ها و Formal Design توسعه میدهد . دیتابیس ها معمولاً از طریق یک DataBase Managemenet System یا DBMS کنترل می‌شوند . دادهها و DBMS و برنامه‌هایی که با آن‌ها مرتبط می‌شوند را Database System گویند .
اگر بخوام خودمونی بگم دیتابیس چیه، میتونم بگم یک سیستم است که دادههایی را میگیرد و آن‌ها را به صورتی سازمان یافته و باقاعده ذخیره میکند و به خاطر اینکه قواعد و فرمول دارند امکان دسترسی به آن‌ها نیز ساده است و میتوان به آن‌ها دسترسی داشت . یعنی دادهها به صورت هردمبیل

ذخیره نمیشوند و مرتب و منظم هستند . هر جا نظم هست خب قاعده و فرمول میشه براش تهیه کرد و به همین خاطر فرمول ها و دستوراتی جهت دسترسی دارند .

به طور کلی دو دسته Database وجود دارد :

۱. SQL
۲. NoSQL

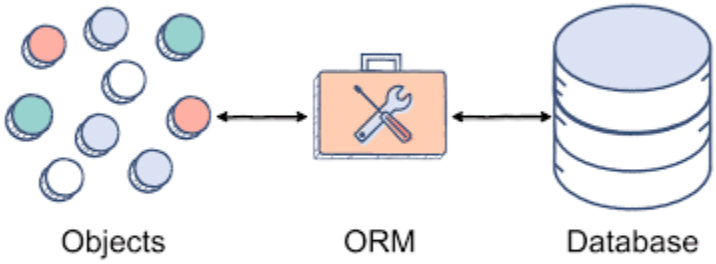


SQL Database چیست ؟ کلمه SQL مخفف Structured Query Language است و به دیتابیس هایی مثل Oracle, PostgreSQL, MySQL, Microsoft SQL Server, SQLite , ... دیتابیس های SQL میگویند . اگه یادتونه گفتیم که دیتابیس ها توسط DBMS یا DataBase Management System کنترل می‌شوند و در دیتابیس های SQL توسط Relational DataBase Management System یا RDBMS مدیریت و کنترل می‌شوند . میگن که این نوع دیتایس ها Vertical Scalable هستند و به صورت عمودی Scale میشند، یعنی دادههای جدید زیر دادههای قدیمی قرار میگیرند و به صورتی عمود و استوار (مرد) هستند . یکی از ویژگی‌های دیگر SQL Database ها Relational بودن انهاست . من تقریباً میتونم بگم که هیچ ایده ای درمورد چیزایی که تا اینجا گفتم ندارم، یعنی نه میدونم DBMS چیه ؟ نه میدونم RDBMS چیه ؟ نه میدونم Vertical Scalable چیه و همچنین نه میدونم Relational بودن چیه ؟ اینا رو به صورتی طوطی‌وار از گوگل اینجا کپی کردم . این درس رو توی دانشگاه داشتیم ولی من تا اومدم بردارم و پاس کنم انصراف دادم :-) اها اینو هم بگم که این نوع دیتابیس ها ساختارمند هستند، ساختار مشخصی دارند، ممکنه مجموعه‌ای از دیتابیس ها داشته باشیم، هر کدام از آن‌ها مجموعه‌ای از Table ها داشته باشند که ممکنه این Table ها با هم ارتباط داشته باشند و ممکنه نداشته باشند، هر Table مجموعه‌ای از Column ها داره، هر Column نوع خاصی از داده رو در خود ذخیره میکنه . سینتکس این دیتابیس ها مشابه هست و همه از سینتکس SQL پشتیبانی میکنند و دستورات در همه یکسان است، مثلاً در همه آن‌ها برای انتخاب

یک داده از دستور SELECT استفاده می‌شود و مثلاً عملیاتهای CRUD در همه یکسان است و فقط ممکن است در برخی از Feature ها تفاوت باشد .

دیتابیس های NoSQL چیستند ؟ NoSQL مخفف Not Only Structured Query Language است . به دیتابیس هایی مثل ,Radis mongoDB, Apache HBASE, cassandra , ... دیتابیس های NoSQL میگن . این دیتابیس ها برخلاف SQL ها non-relational هستند و به صورت Horizontal Scalable می‌باشند . گفتیم توی SQL Database ها RDBMS اونها رو مدیریت و کنترل میکنه ولی توی NoSQL ها هر دیتابیس متفاوته، یکیش CLI داره مثل Radis، یکیش ممکنه GUI داشته باشه مثل mongoDB و ... یعنی NoSQL تقریباً مخالف SQL است . گفتیم که SQL دیتابیس ها ساختارمند هستند و همه اونها جدا از اینکه چه نوع SQL دیتابیسی هستند تقریباً ساختار یکسانی دارند و قواعد و قوانین یکسان، اما در NoSQL ها چنین نیست ! هر کدام از آن‌ها با دیگری متفاوت است، ممکن است یکی از آن‌ها مثل Radis دادهها رو به صورت Key-Value ذخیره کند، یکی از آن‌ها دادهها را در Graph، یکی در فایل و Document و یکی هم به صورت Column Store . برخلاف SQL دیتابیس ها NoSQL ها دستورات یکسانی ندارند، اصن هر طوری دلشون خواسته تعریف کردن، یه انارشی عجیبی حس میکنم بینشون، شاید چون تاحالا کار نکردم چنین احساسی دارم و باید دخیل شم تا بفهمم .

ORM چیست ؟ ORM مخفف Object-Relationa Mapping است . در گذشته در زبانهای برنامه نویسی اگر قصد ارتباط با Database را داشتند، از یک کتابخانه جهت ایجاد Connection استفاده میکردند و دستورات SQL خود را به صورت خام از طریق آن به RDBMS میدادن تا در دیتابیس اعمال کند . این ارتباط مستقیم با RDBMS و وارد کردن دستورات خام SQL در کد موجب بروز مشکلات امنیتی مثل SQL Injection شد و همچنین برنامه نویس باید برای هر چیزی Query مینوشت و سرعت توسعه به شدت پایین بود و حتی میتونم بگم که اگر بلد نبود Query تروتمیز بنویسه باعث کاهش سرعت اپلیکیشن میشد . حفره امنیتی SQL Injection یکی از قدیمی ترین حفرات امنیتی است که هنوز که هنوز است شما میتونید در بسیاری از اپلیکیشن ها پیدا کنید . یعنی کاربر که داشت با اپلیکیشن کار میکرد، چون دستورات SQL مستقیم و خام در اپلیکیشن وجود داشت، سعی میکرد با وجودی هایی آن‌ها را به نفع خود تغییر دهد و به جای مثلاً گرفتن پست های ذخیره شده در Database، کاربران ذخیره شده را بگیرد . این حفره امنیتی رو به صورت کامل در آینده بررسی خواهیم کرد و طریقه رفع اون رو خواهیم آموخت . برای اینکه این مشکلات امنیتی ایجاد نشود اومدند و ساختاری تحت عنوان ORM ایجاد کردن تا ارتباط مستقیم اپلیکیشن با دیتابیس رو از بین ببرند و یک واسطی مابین آن‌ها قرار بدهند که جلوی SQL Injection ها رو بگیره و اینو هم بگیم که بسیار موفق بود و میزان این حفره امنیتی رو به صورت چشم گیری کاهش داد . مثلاً در یک فریمورک مثل Laravel که با زبان PHP نوشته شده و طراحی وبسایت رو به یه کار خیلی ساده تبدیل کرده یک ORM به نام Eloquent ORM وجود دارد . یا در فریمورک Django که به زبان پایتون نوشته شده است و کار طراحی وبسایت با پایتون رو بسیار دلنشین کرده هم ORM خودش رو داره که فک کنم اسمش sqlalchemy باشه . البته میگن ASP چنین ساختاری نداره و هنوز هم که هنوز هست برنامه نویساش Query های SQL رو مستقیماً وارد میکنن . پس اگه یه روز (بری سفر) یه جایی خواستید یک وبسایت ASP رو Pentest کنید قطعاً دنبال SQL Injection باشید . اما این یارو ORM درسته که بسیاری از حفرات SQL Injection رو درست کرد، سرعت توسعه رو افزایش داد، Query ها رو ترو تمیز تر کرد و سرعت اجراشون رو زیادتر ولی خب هرچیزی که جدید میاد باگ داره و میتونه حفره امنیتی ایجاد کنه که کرد، ORM وامد SQL Injection رو رفع کنه و خودش ORM Injection بوجود آورد و هکرها با فهمیدن ساختار ORM تونستن کدهای SQL رو که ORM اجرا میکرد به نفع خودشون تحریف کنن و دستورات SQL خودشون رو توسط ORM اجرا کنن .



MySQL چیست ؟ دیتابیسی که ما اینجا قرار هست ازش استفاده کنیم MySQL هست که یک DataBase Managemenet متن باز رایگان و RDBMS هست . برای نصبش کافیه که اگه ویندوزی هستید Wamp, Xampp رو نصب کنید .

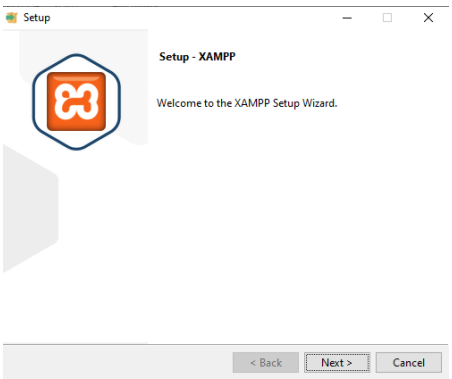
لینک دانلود XAMPP اینجاست : https://www.apachefriends.org/download.html

لینک دانلود WAMP اینجاست : https://www.wampserver.com/en

XAMPP و WAMP پکیج نرم افزاری هستند که براتون PHP, MySQL, Apache (البته XAMPP علاوه بر اینها Perl رو هم نصب میکنه) رو نصب میکنن و میتونین باهاش به طراحی وبسایت بپردازید . XAMPP نسخه ویندوزی و لینوکسی و مک رو داره و WAMP فقط مخصوص ویندوز هست . اینو هم بگم که اگه خواستید روی لینوکس این پکیج رو بدون استفاده از XAMPP نصب کنید میتونید به لینک :

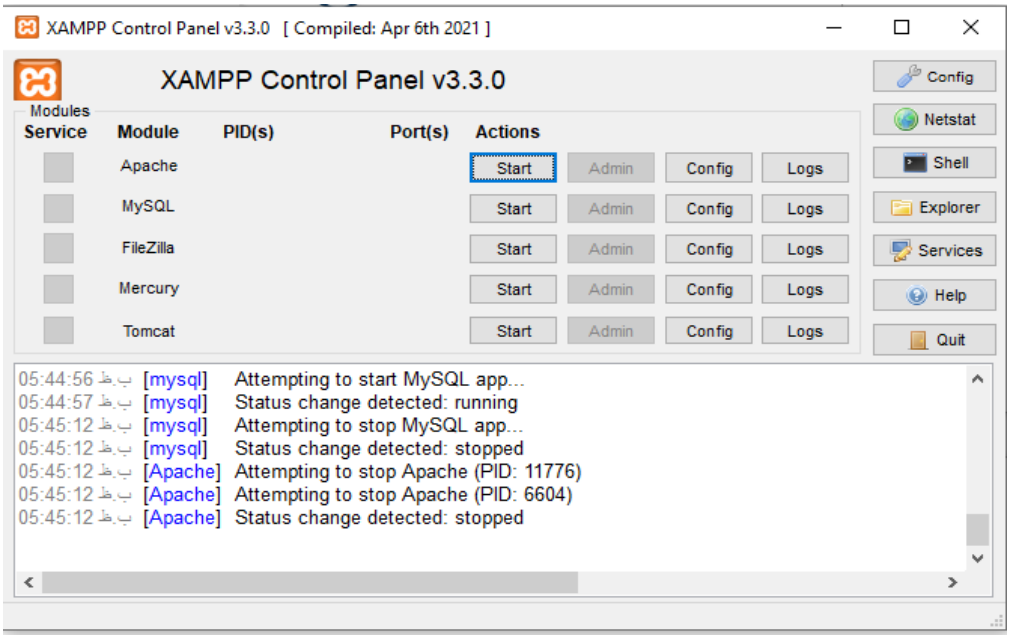
https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-۲۲-۰۴ مراجعه کنید و خب نحوه نصب رو به صورت کامل توضیح داده است .

من XAMPP روی ویندوز نصب میکنم و خب از نصبش چیزی نمیگم چون چندتا Next داره و درنهایت Finish .

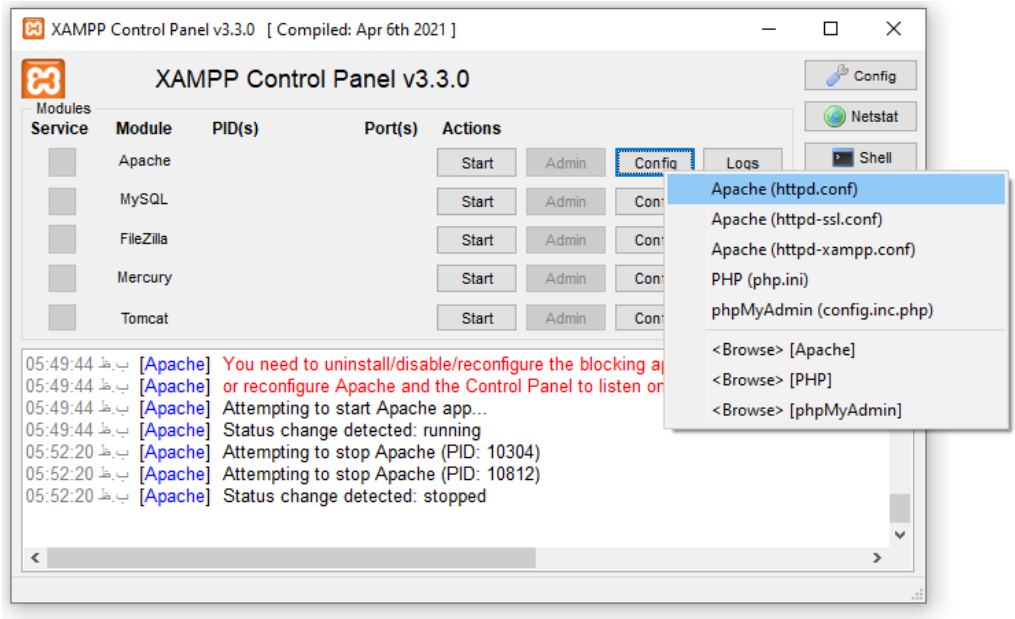


وقتی XAMPP رو نصب کردید با اجرای آن Control Panel آن را خواهید دید . در این کنترل پنل برای اجرای MySQL باید دکمه Start جلوی آن را بزنید تا رنگش سبز شود و به معنی استارت شدن است . اگر بخواهید کدهای PHP رو هم اجرا کنید دکمه Start جلوی Apache رو هم بزنید

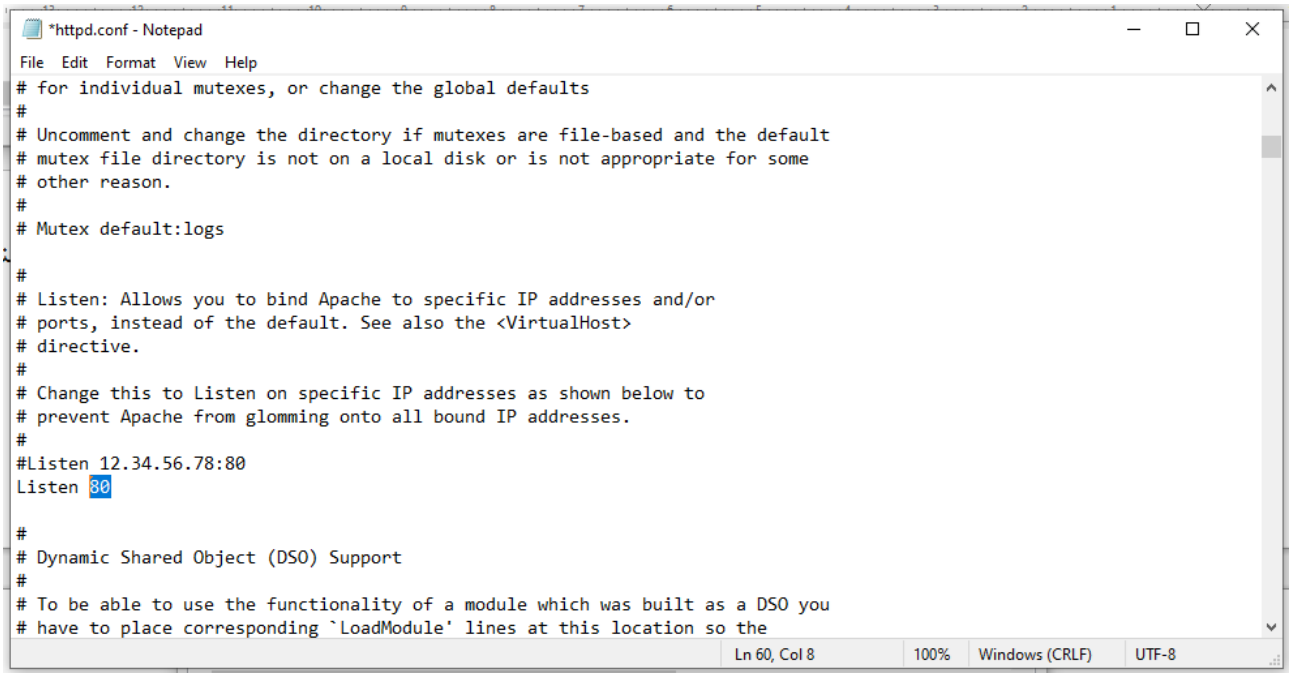
تا اپاچی هم رنگش سبز بشه . اگر Apache اجرا نشد به معنی این است که ممکن است پورت ۸۰ و ۴۴۳ که توسط Apache و کلاً Web Server استفاده می‌شود توسط یک پروسه دیگه در حال استفاده است و بایستی آن پروسه رو ببندید و سپس اقدام به Start کنید .



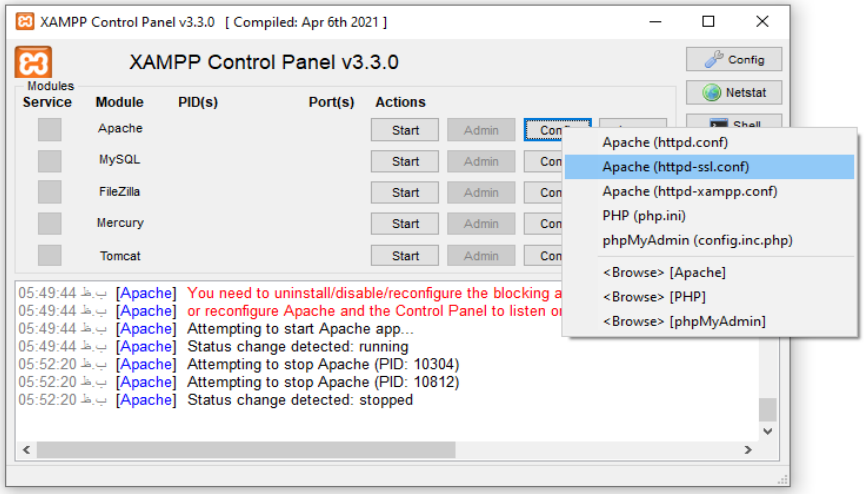
اگر Apache اجرا نشد و شما هم نتونستید پورت های ۸۰ یا ۴۴۳ که توسط پروسه دیگری آشغال شده‌اند رو به Apache بدید میتونید این پورتها رو توی پیکربندی های Apache تغییر بدید . اولین فایل پیکربندی httpd.conf است که باید پورت ۸۰ داخل اون رو به چیزی که میخواید تغییر بدید :

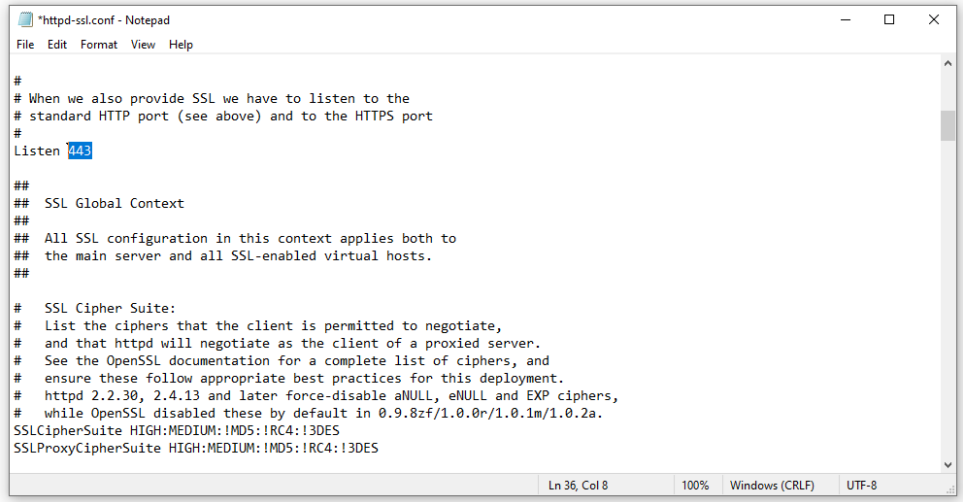


وقتی بر روی این گزینه کلیک کنید یک notepad برای شما باز میشه که میتونید پورت ۸۰ رو عوض کنید :

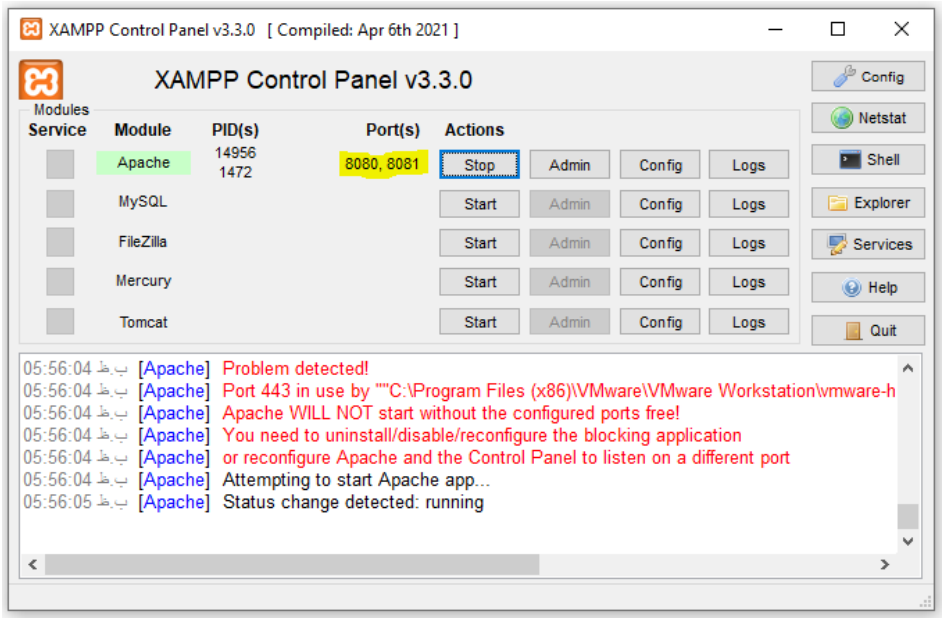


دومین پیکربندی مربوط به پورت ۴۴۳ است که میتونید اون رو از گزینه زیر ببینید و تغییر دهید :

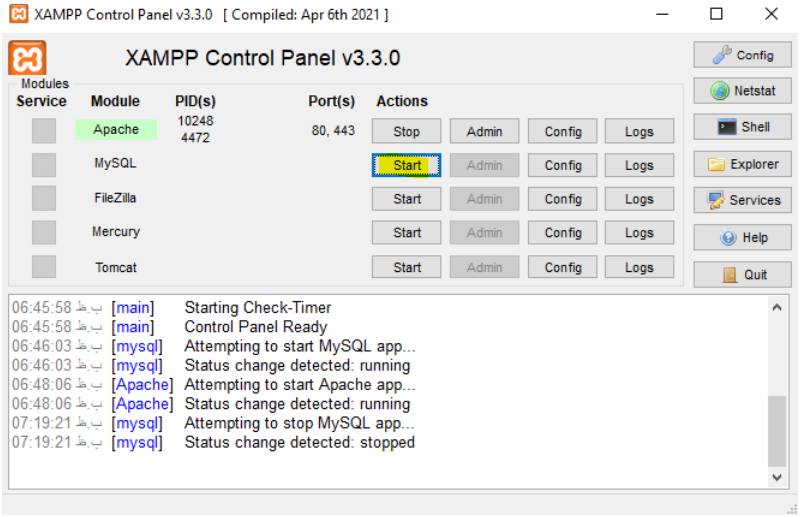




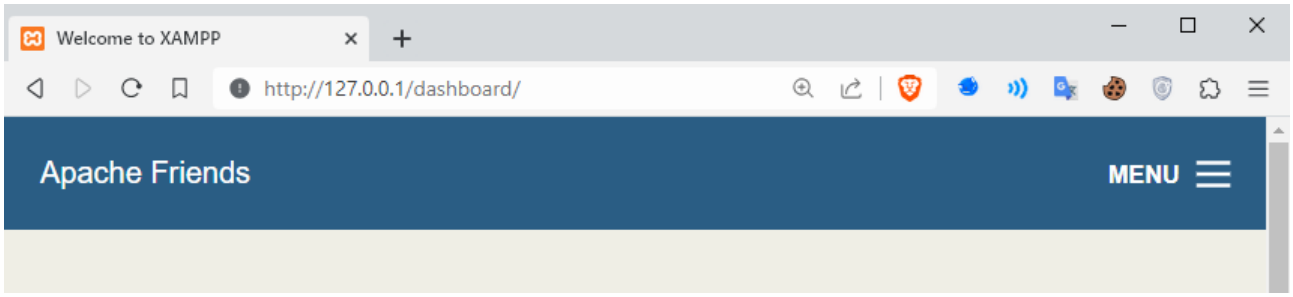
خب من پورت ۸۰ رو به ۸۰۸۰ و پورت ۴۴۳ رو به ۸۰۸۱ تغییر دادم و Apache اجرا شد :



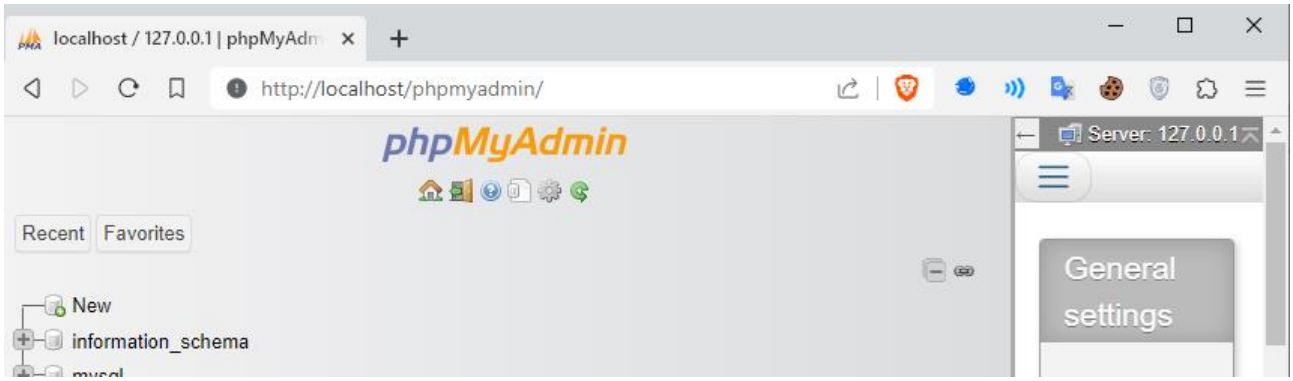
و اگر هم میخواهید که MySQL رو اجرا کنید کافیه دکمه Start جلوی آن را بزنید تا اجرا شود :



حال که نصب کردیم و Start یه تست بزنیم ببینم کار میکنن یانه ؟ برای تست کردن Apache کافیه که localhost یا ۱۲۷.۰.۰.۱ رو داخل مرورگر باز کنیم تا ببینم میاره یا نه :



محتوایی که در تصویر قبل دیدیم مربوط به دایرکتوری C:\xampp\htdocs\dashboard است . یعنی آدرس localhost درواقع فایل index.html این دایرکتوری رو اجرا میکنه . اگه هم پورت رو عوض کرده بودید کافیه که جلوی ۱۲۷.۰.۰.۱ به شکل [PORTNUMBER]: بنویسید . حال نوبت بررسی MySQL هست که آدرس localhost/phpmyadmin رو وارد کنیم و اگر لود شد یعنی درست است :



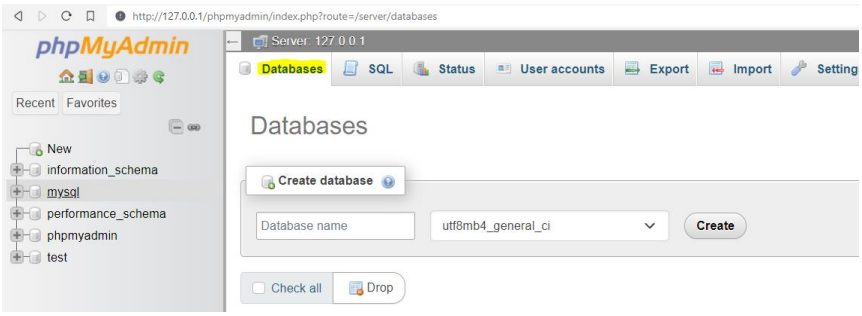
رابطی که ما ازش برای پیکربندی دیتابیس ها و ایجاد Table ها و Column ها استفاده میکنیم phpmyadmin هست و عموما هم بر روی وبسایت ها این برنامه به عنوان رابط با MySQL استفاده میشه .

اجزای سازنده DataBase ها چیستند ؟ یک DataBase که SQL است از اجزایی ساخته می‌شود که به عبارت زیرند :

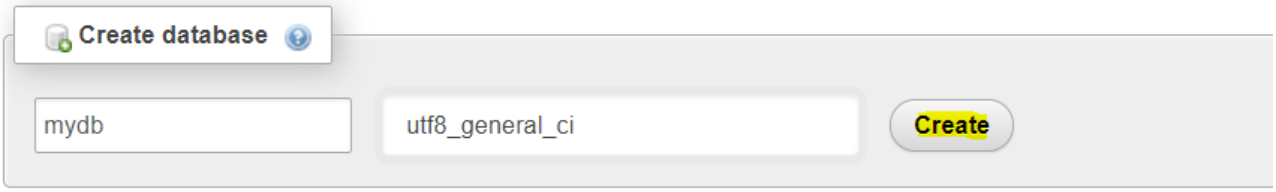
۱. Table ها
۲. Column ها
۳. Row (Record) ها
۴. Data Value ها

یک دیتابیس شامل چندین Table می‌باشد که در هر Table اطلاعات مربوط به چیزی ذخیره می‌شود و Column, Row, Data Value از اجزای Table ها هستند .

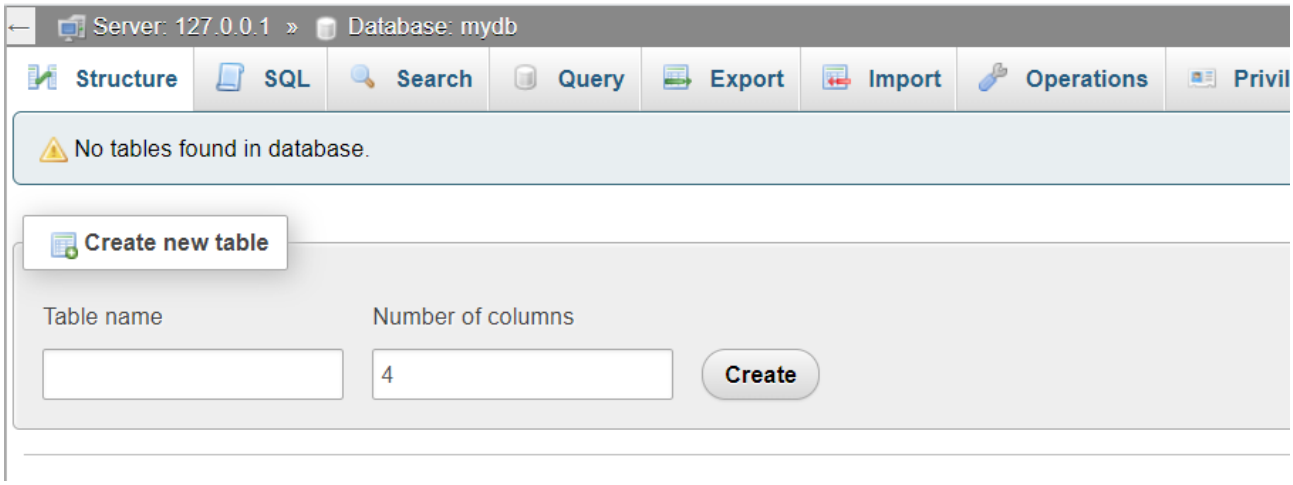
چطوری توی phpmyadmin یک پایگاه داده ایجاد کنیم ؟ برای ایجاد یک پایگاه داده در phpmyadmin کافیه که از منوی بالا گزینه Databases رو انتخاب کنیم .



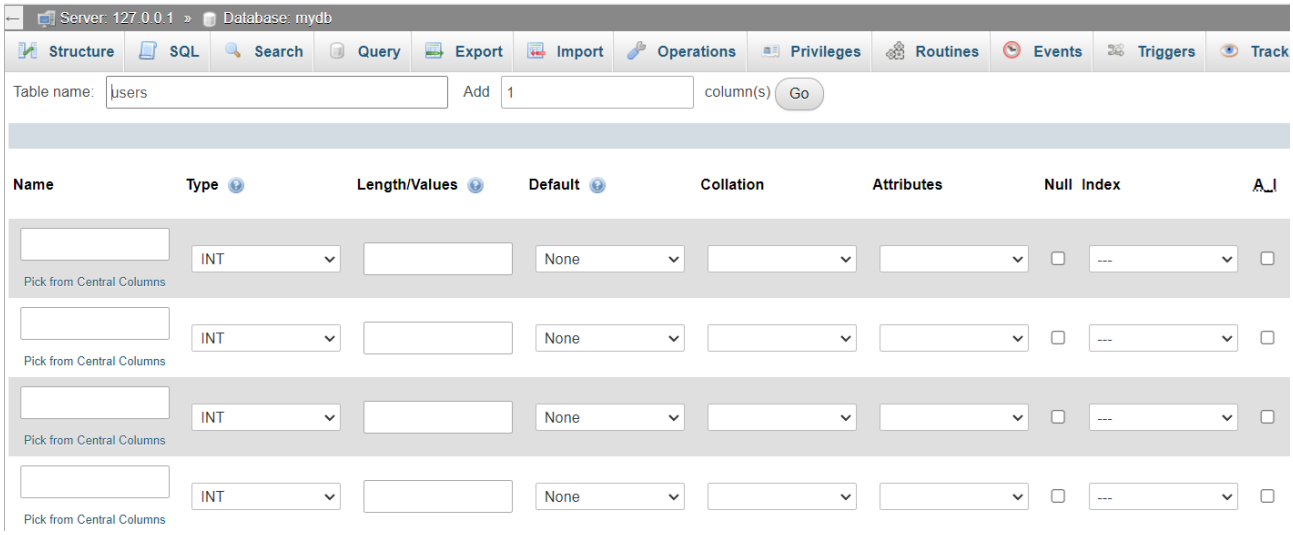
در این منو شما لیست Database های موجود را خواهید دید و از قسمت Create database میتونید اقدام به ساخت یک پایگاه داده کنید، نام آن را وارد کنید و همینطور Collation آن را utf۸_general_ci قرار دهید تا هم فارسی و هم انگلیسی را پشتیبانی کند و دکمه Create رو بزنید :



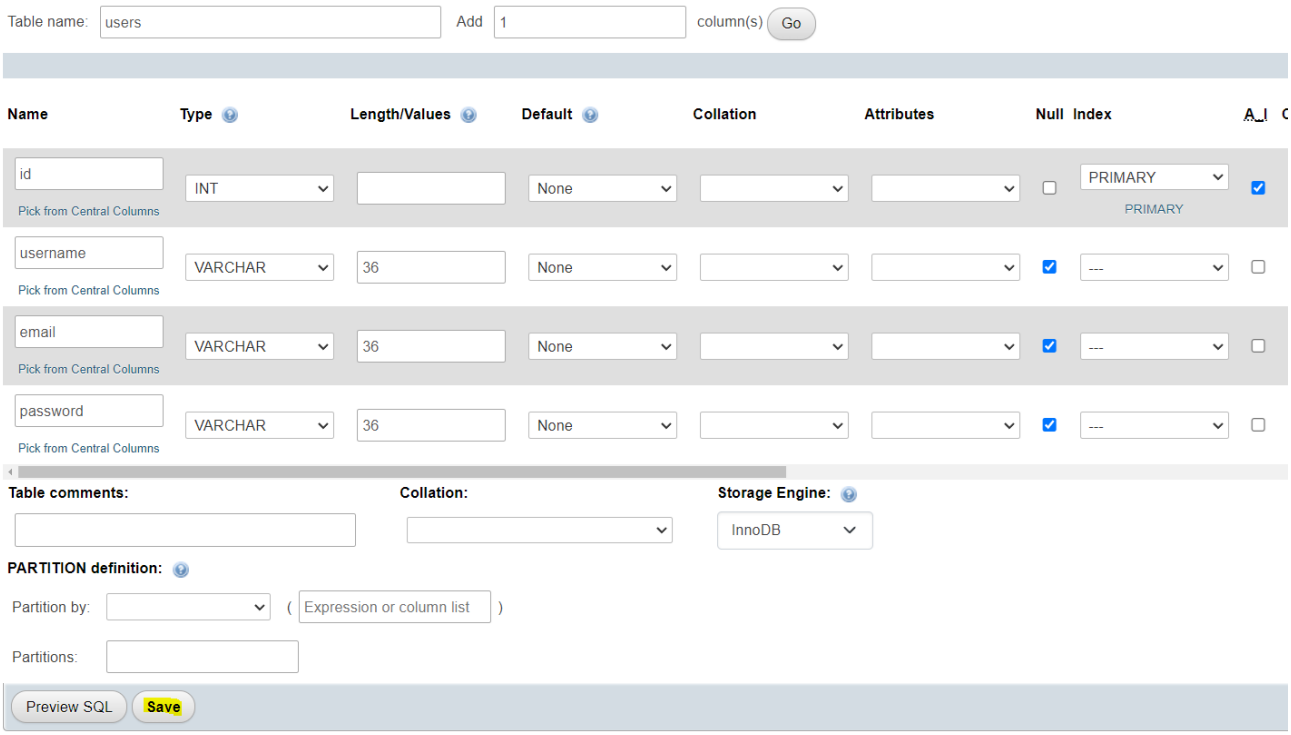
خب بعد از این کار DataBase ساخته میشه و با صفحه زیر روبرو میشیم :



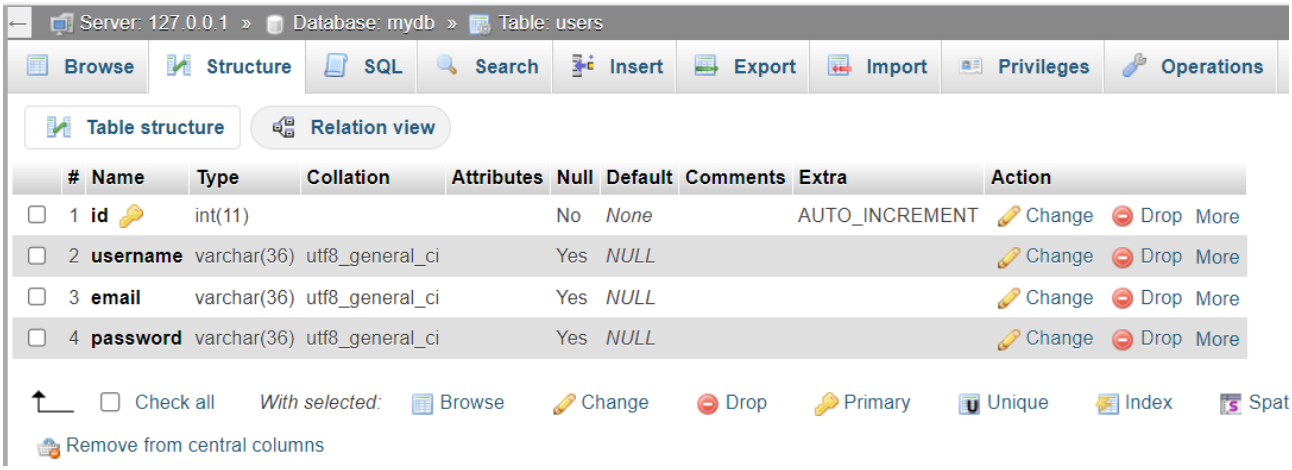
توی اینجا ما Table های داخل دیتابیس رو میسازیم، Table Name و بعد تعداد Column های داخل Table رو مشخص کنید و دکمه Create رو بزنید .



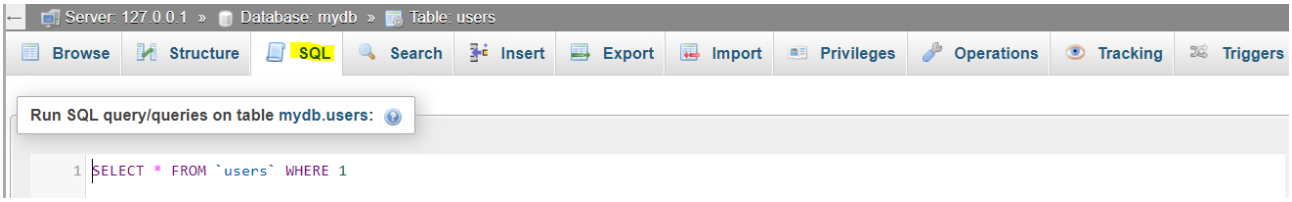
توی صفحه بالا باید مشخصات Column ها رو وارد کنیم . Name، Type و ... رو . جزئیات اینا توی حوصله این فایل نمیگنجه و خودتون باید برید و درموردش بخونید . توی سایت W۳Schools درمورد SQL اموزشای خوبی پیدا میشه . من اطلاعات چهار Column رو وارد کردم و بعد روی دکمه Save میزنم تا ذخیره بشه :



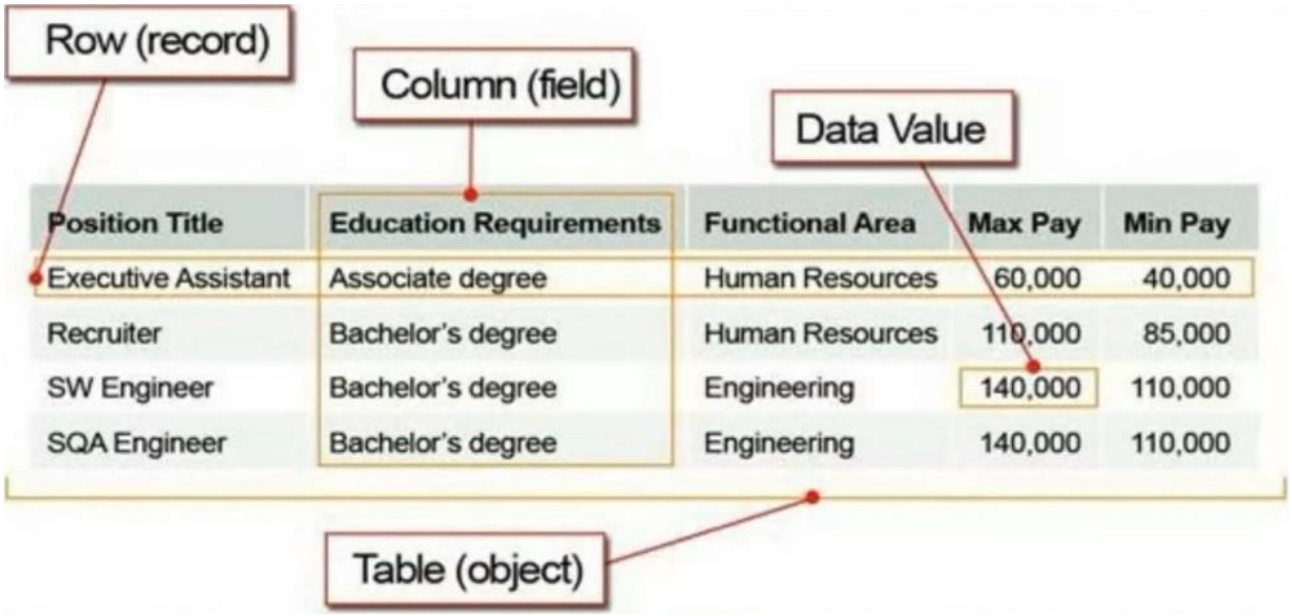
در تصویر زیر ساختار Table من رو میبینید که شامل چه Column هایی با چه Type هایی هست :



برای اینکه بتونیم دستورات SQL رو روی این جدول تست کنیم میتونیم از منوی بالا گزینه SQL رو انتخاب کنیم تا به صفحه زیر بریم :



توی این صفحه میتونیم دستورات SQL رو وارد کنیم و با زدن Ctrl+Enter اون رو بر روی Table خودمون اجرا کنیم . دقت کنید که تمام کارهایی که تا اینجا انجام دادیم مثل ساختن دیتابیس، ساختن Table و ایجاد Column ها بدون دستورات SQL بوده است و همه اینکارها را میتوان با دستورات SQL انجام داد . درواقع phpmyadmin دستورات من رو به SQL تبدیل کرده و در DBMS اجرا کرده . جزئیات ساختار یک Table را میتوان در تصویر زیر دید :



Position Title	Education Requirements	Functional Area	Max Pay	Min Pay
Executive Assistant	Associate degree	Human Resources	60,000	40,000
Recruiter	Bachelor's degree	Human Resources	110,000	85,000
SW Engineer	Bachelor's degree	Engineering	140,000	110,000
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000

گفتیم یک دیتابیس شامل چندین Table است و هر Table شامل چندین Column یا Field . در هر Table دادهها در قابل Row یا Record وجود دارند . هر Row شامل چندین داده است که هر کدام را Data Value یا Cell میگویند و هر کدام از آن‌ها در مربوط به یک Column است . اگر بخوام باز ذکر مثال بگم : در تصویر بالا ما ۵ ستون یا Column داریم که عبارت اند از : Position Title, Education Requirements, Functional Area, Max Pay, Min Pay .

یکی از دادههایی که در این جدول ذخیره شده است مربوط به Position Title برابر Executive Assistant است . Educational Requirements این Executive Assistant برابر Associate Degree است . Functional Area آن برابر Human Resources است و Max Pay آن ۶۰۰۰۰ و Min Pay آن ۴۰۰۰۰ . دادهها در DataBase های SQL به این شکل Organized میشن .

DataBase Server چیست ؟ یک سخت‌افزار است که یک نرم‌افزار Database را اجرا میکند . Database Server سرویس پایگاه داده را ارائه میکند و میتواند شامل چندین Database مختلف باشد . هر Database میتواند مربوط به یک وبسایت جداگانه یا یک بخش جداگانه از یک سازمان باشد که هر Database نیز شامل جداولیست و در هر جدول اطلاعاتی ذخیره شده است .

CRUD چیست ؟ ما دریک چهار عمل اصلی داریم که میتونیم بر روی یک Database اجرا کنیم . این چهار عمل اصلی عبارت اند:

۱. Create :: INSERT
۲. Read :: SELECT
۳. Update :: UPDATE
۴. Delete :: DELETE

به این چهار عمل اصلی به صورت مخفف CRUD میگویند . Create همان وارد کردن اطلاعات به دیتابیس است که با دستور INSERT انجام می‌شود . Read یعنی خواندن دادهها که از طریق دستور SELECT انجام می‌شود و Update برای تغییر دادههای موجود با استفاده دستور UPDATE و همچنین Delete منظور حذف کردن دادههای موجود است که با دستور DELETE انجام می‌شود .

دقت کنید که SQL به حروف بزرگ و کوچک حساس نیست، یعنی میتونید دستورات رو با حروف بزرگ و یا کوچک وارد کنید ولی پیشنهاد این است که دستورات اصلی SQL مثل INSERT, SELECT, DELETE, UPDATE و ... رو با حروف بزرگ بنویسیم هر چند با حروف کوچک هم کار میکنند .

دستور INSERT چگونه میتواند Record جدید به Table ما وارد کند ؟ سینتکس دستور INSERT به صورت کلی به شکل زیر است :

```
INSERT INTO TABLE_NAME (COL1, COL2, COL3, ...) VALUES (VAL1, VAL2, VAL3, ...)
```

فرض کنید یک Table داریم که شامل Column هایی به شرح زیر است :

۱. username
۲. email
۳. password

اگر بخواهیم از طریق دستور INSERT به این جدول یک Record اضافه کنیم باید به شکل زیر عمل کنیم :

```
INSERT INTO `users` (`username`, `email`, `password`) VALUES ("theSecDude", "thesecdude.dev@gmail.com", "123456")
```

میبینید که نام جدول و ستون‌ها رو در `` قرار دادم و اگر هم قرار ندیم مشکلی ایجاد نمیشه . ولی خب ممکنه یکی از کلمات کلیدی SQL رو ما به عنوان نام یک ستون تعریف کرده باشیم و بهتر است که اونها توسط `` از کلمات کلیدی جدا کنیم و اینکه نمیتونید نام Tabel و ستون‌ها رو با "" یا '' نشون بدید و ارور میده و فقط Value ها رو میتونید چنین کنید .

```
INSERT INTO users (username, email, password) VALUES ("JAFAR", "jafar@gmail.com", "123456")
```

نکته امنیتی که اینجا وجود داره این هست که Password در دیتابیس ها نباید به صورت Plain Text ذخیره شوند و حتماً باید قبل از ذخیره شدن توسط الگوریتم های رمزنگاری و Hash هایی مثل MD۵, SHA۲۵۶ و ... رمزنگاری شود و بعد در دیتابیس ذخیره شوند . اینکار به این علت است که اگر یه وقتی یک هکر به دیتابیس دسترسی پیدا کرد نتواند Credential های کاربران و اطلاعات حساس رو بخونه . شاید بگید خب اگه Hash بشه پس چطوری کاربر بتونه Login کنه ؟ خب اطلاعات کاربر در صفحه لاگین وبسایت وارد میشه و Back-END وبسایت رمز وارد شده کاربر رو Hash میکنه و سپس این Hash رو با Hash داخل دیتابیس مقایسه میکنه و اگر برابر بود کاربر وارد میشه . توابع Hash هر ورودی را بگیرند یک مقدار را برای آن خارج میکنند . مثلاً اگر من ۱۲۳۴۵۶ رو امروز به تابع بدم یک مقدار به من میدهد و اگر ۱۰۰ روز دیگر هم به آن تابع ۱۲۳۴۵۶ را

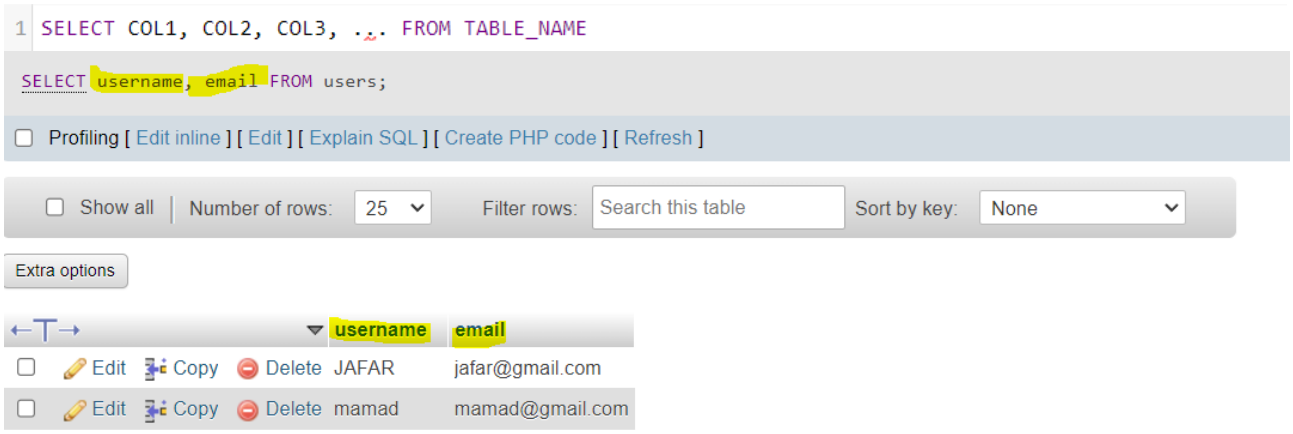
بدم همین مقدار را به من برمیگرداند و همچنین یکی دیگر از ویژگی‌های Hash ها یکطرفه بودن و غیر قابل بازگشت بودن انهاست . میتونید الگوریتم های آن‌ها را در گوگل جستجو کنید .
الگوریتم های Hash زیادی وجود دارند مثل MD۵ که در گذشته استفاده میشدند و امروزه دیگر پیشنهاد نمیشوند چرا که ضعیف محسوب می‌شوند و الگوریتم هایی مثل SHA۱, SHA۲۵۶ و ... که امروزه بسیار مورد استفاده قرار میگیرند .

دستور SELECT در SQL چه میکند ؟ برای خواندن دادههای داخل یک جدول از این دستور استفاده می‌شود . سینتکس استفاده از این دستور به شکل زیر است . اگر بخواهیم مقادیر همه ستون‌های داخل یک جدول رو ببینیم به شکل زیر عمل میکنیم :

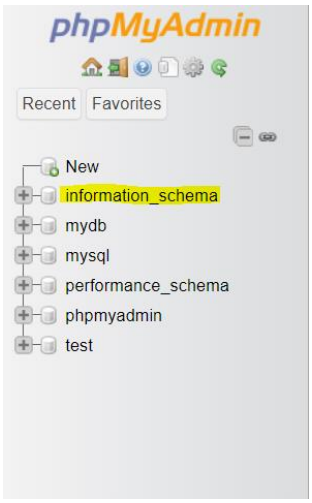
```
1 SELECT * FROM TABLE_NAME
```

و اگر بخواهیم به مقادیر ستون‌های خاصی از جدول دسترسی پیدا کنیم باید آن ستون‌ها را بنویسیم، به شکل زیر :

```
1 SELECT COL1, COL2, COL3, ... FROM TABLE_NAME
```



راستی به صورت پیش‌فرض چند پایگاه داده بر روی MySQL وجود دارند که در تصویر زیر میبینید . دیتابیس information_schema رو به یاد داشته باشید چرا که در Exploit کردن باگهای SQL Injection اهمیت دارد .



Update کردن یک Record در یک جدول با دستورات SQL چگونه است ؟ دستوری در SQL وجود دارد به نام UPDATE که این کار رو برای ما انجام میدهد . سینتکس این دستور به شکل زیر است :

```
UPDATE `TABLE_NAME` SET COL1='NEW_VALUE', COL2='NEW_VALUE', COL3='NEW_VALUE' WHERE COLX="VALUE"
```

کلمه WHERE ممکن است توجه جلب کرده باشد و به منظور ایجاد شرط استفاده می‌شود . برای UPDATE کردن ما باید یک Record رو داشته باشیم تا UPDATE شود و به همین منظور نوشتیم در جدول TABLE_NAME جایی که COLX برابر است با VALUE . به مثال زیر هم توجه کنید :

```
UPDATE `users` SET `password` = 'ABCDEFGH' WHERE `id` = 2;
```

نکته‌ای که باید نسبت به آن آگاه باشیم این هست که دستورات SQL نسبت به خط بعد اصلاً حساسیتی ندارد و کافیست هر وقت که نیاز شد با یک ; دستور رو از دیگر جدا کنیم . این موضوع در مبحث حمله و امنیت هم اهمیت پیدا میکنه و مثلاً مهاجم میتونه از طریق CSRF Injection دستوراتش رو خط به خط بنویسه و به سمت وب سرور بفرسته و SQL Injection بزنه، در آینده با آن آشنا می‌شویم .

```
UPDATE `users`
SET
`password`="ABCDEDFFASDADASDASD"
WHERE
`id`=3; UPDATE `users` SET `password`="1221343234534" WHERE `id`=2;
```

حذف کردن یک Record از جدول با دستورات SQL چگونه انجام می‌شود ؟ برای حذف کردن یک Record باید از دستور DELETE استفاده کنیم . سینتکس این دستور به حالت کلی به شکل زیر است :

```
DELETE FROM `TABLE_NAME` WHERE `COLX`="VALUE";
```

دقت شود که شرط ما باید یک Record رو انتخاب کنه نه چند Record رو . یعنی تنها یک Record باشد که COLX آن برابر VALUE باشد . مثلاً در زیر نمونه‌ای از این دستور رو میبینیم :

```
DELETE FROM `users` WHERE `id`=3;
```