# საქართველოს იუსტიციის სამინისტრო

## კიბერ უსაფრთხოების საინფორმაციო ბიულეტინი
## პოლიტიკა: ვებ საიტების ოპტიმიზაცია

*IT ინფრასტრუქტურის დეპარტამენტი*

## სარჩევი:

# შესავალი

ვებ საიტების კოდის ოპტიმიზაცია წარმოადგენს მნიშვნელოვან ფაქტორს საიტების ჰოსტინგ ინფრასტრუქტურის გამართული და ეფექტური მუშაობისთვის. არაოპტიმიზირებულმა და ცუდად დაწერლმა საიტის კოდმა  შეიძლება გამოიწვიოს მთელი რიგი პრობლემები ვებ საიტების მუშაობის დროს, რომლებიც ქვემოთ იქნება ჩამოთვლილი.  ყველა პირი ვინც მონაწილეობას ღებულობს ჰოსტინგ ინფრასტრუქტურაში არსებული საიტების შექმნაში, მოდიფიცირებაში, მხარდაჭერაში, ან ვებ სერვისების უზრუნველყოფაში, ვალდებულია დაიცვას ქვემოთ ჩამოთვლილი პირობები და პასუხისმგებელია მის მიერ შესრულებულ სამუშაოზე.

## 2. პოლიტიკის მიზნობრიობა

პოლიტიკის მიზანია დადგინდეს ვებ საიტის მკაცრად განსაზღვული სხვადასხვა ტიპის კოდების ოპტიმიზაციის თავარი პრინციპები, რომლის მიხედვითაც დამზადდება/ შეიმნება ახალი ვებ საიტი, ან მოდიფიცირდება არსებული. რადგანაც არაოპტიმიზირებული ვებ გვერდების ინფრასტრუქტურაში განთავსების შემთხვევაში არსებობს შემდეგი პრობლემები:

- ვებ საიტის შენელებული მუშაობა;
- სერვერულ – ქსელური რესურსების არაეფექტური გამოყენება ან მათი გადავსება;
- მომხმარებლის ბროუზერის შენელება გვერდის გახსნის დროს „გაჭედვა“;
- ქსელის სტრაფიკის გაზრდა ;
- აპარატურული რესურსების გადავსების გამო საიტის „გაჩერება“
- საძიებო სიტემების მიერ არასწორი ინდექსირება;
- ვებ ინფრასტრუქტურის მართვის გართულება;
- მზარდი კიბერ საშიშროების რისკი (საიტების ზრდასთან ერთად);
- ჰოსტინგ ინფრასტრუქტურის მოცულობის გაზრდა;

## 3 ვის ეხება აღნიშნული პოლიტიკა?

აღნიშნული საინფორმაციო ბიულეტინი ეხება ყველა იმ დეველოპერს, რომელიც მონაწილეობას ღებულობს საქართველოს იუსტიციის სამინისტროს მიერ დაკვეთილ ვებ საიტის/რესუსრის შექმნაში, ან არსებული საიტის მოდიფიკაციასა და პერიოდულ განახლებში, რომლებიც განთავსდება ამავე სამინისტროს დაცულ ვებ ჰოსტინგ ინფრსტრუქტურაში.

## 4. კონფიდენციალურობა და უსაფრთხოება

აღნიშნული პოლიტიკის მიხედვით დადგენილი პირობები აუდიტირდება პერიოდულად.

აღნიშნული პოლიტიკის დარღვევის და მისი უგულველყოფა-არშესრულების დროს საქართველოს იუსტიციის სამინისტროს შეუძლია მიმართოს შიდა განაწესით მიღებულ ზომებს.

# 5.round-trip (RTT)*

## 5.1. დროის შემცირება.

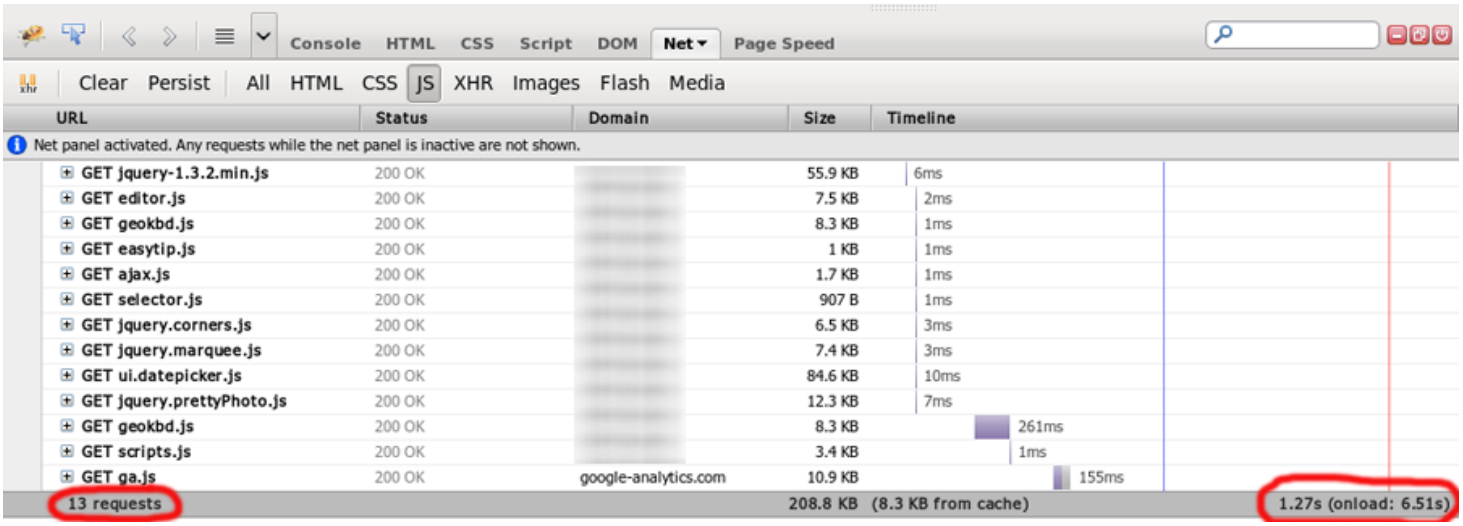
დროის მონაკვეთი, როდესაც კლიენტის მოთხოვნას პასუხობს სერვერი ინფორმაციის მიმოცვლამდე, ამ დროის მონაკვეთში  იგულისხმება DNS lookup-ი.

dns looklup ინდა იყოს მაქსიმალურად შემცირებული, ამასთანავე  ყველგან უნდა იყოს გამოყენებული გზები(path) მისამართების(URL)  მაგივრად, ზედმეტი looklup –ის შესამცირებლად მაგ: სურათის წყარო HTML დოკუმენტში უნდა იყოს /res/images და არა http://site.ge/res/images

## 5.2. JavaScript  კომბინირება

**HTML**  გვერდზე რამოდენიმე **JavaScript**  ერთდროულმა ჩატვირთვამ შეიძლება წარმოქმნას შემდეგი პრობლემები:  ბროუზერების უმეტესობა თანმიმდევრობით ელოდება სანამ JavaScript ფაილი ჩაიტვირთება და „გაიპარსება“,  მანამდე შეაჩერებს სტილების ან გარე ობიექტების ჩატვირთვას, ხშირია შემთხვევები, როდესაც  საიტის ძირითადი ჩონჩხი იტვირტება 5–ზე მეტ წამიანი დაყოვნების შემდეგ.

სურ.1



სურათზე მოცემულია საიტის მაგალითი სადაც **JavaScript** ფაილები არ არის კომბინირებული და საჭიროა 13 http კავშირის დამყარება სერვერთან ვებ გვერდის სრულად ჩასატვირთად, სურათზე ასევე ნაჩვენებია, რომ საიტის ჩონჩხის ჩატვირთვიდან 5.1 წამის (6.51/1.27) განმავლობაში გვერდი აგრძელებედა ჩატვირთვას კლიენტის ბროუზერში.

1. ასეთი შემთხვევების თავიდან ასაცილებლად საჭიროა, JavaScript ფაილების კომბინეირება 1 ფალიში (ან მაქს 3);
2. defer loading – ტქნიკის გამოყენება (როდესაც იტვირთება ისეთი javascript ფუნქციები, რომლებიც არ არის საჭირო ამ გვერდის ფუნქციონირებისათვის, როდესაც ასეთი ფუნქციების რაოდენობა აღწევს 25 ან მეტს საჭიროა ამ ფუნქციების სხვა ფაილში გადატანა და მისი გამოძახება საჭიროებისამებრ. );
3. მთავარი javascript სცენარის ჩატვირთვა ვებ გვერდის გასახსნელად ბროუზერში, შემდეგ მეორე შედარებით მძიმე javascript–ის ჩატვირთვა;
4. javascript ფაილების `<head>` სექციაში გადატანა;
5. javascript ოპტიმიზაცია და ზომის მაქსიმალურად შემცირება
6. მცირე ზომის, არა ქეშირებადი javascript –ი გაწერილი უნდა იყვეს HTML კოდში და არ ცალკე ფაილად. (საუბარია მაქს 10 სტრიქონიან ჯავასკრიპტზე, რომლეთა რაოდენობა რ აუნდა აღწევდეს 3–ს).

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი საიტის შექმნის ან მოდიფიცირების დროს.

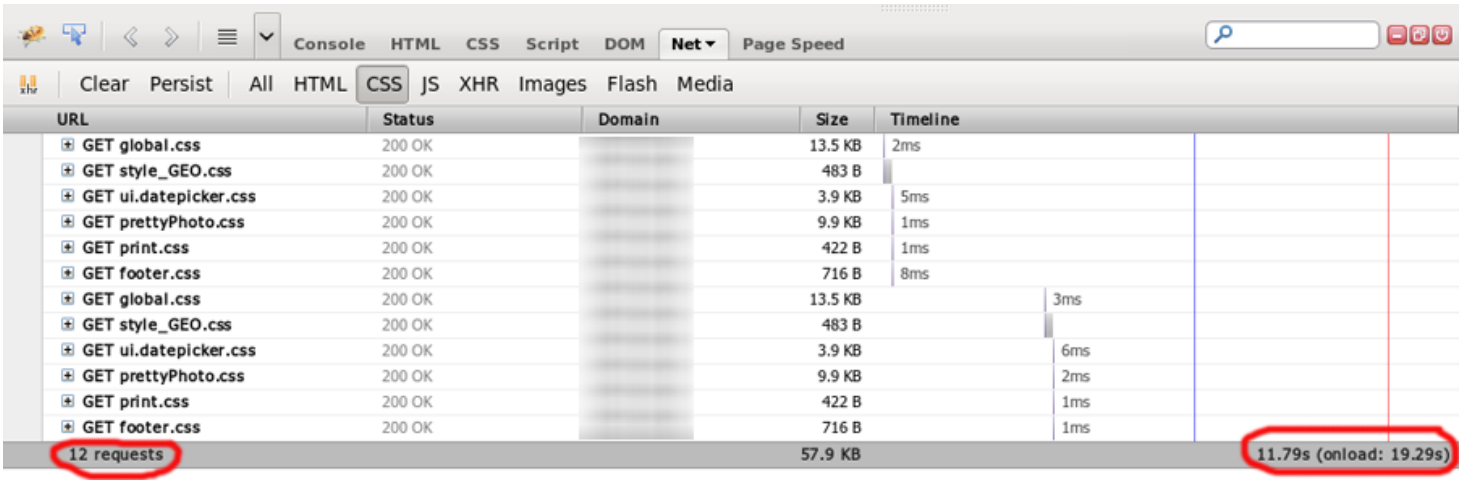
## 5.3. CSS კომბინირება

**HTML** –ში ისევე როგორც JavaScript–ის დროს, გვერდზე რამოდენიმე **CSS** სტილების ერთდროულმა ჩატვირთვამ შეიძლება წარმოქმნას შემდეგი პრობლემები:

1. RTT დროის მონაკვეთის გაზრდა;
2. საიტის ძირითადი ჩონჩხის დაყოვნებითი ჩატვირთვა;

3. კლიენტის ბროუზერის შენელებული მუშაობა (სტრიქონების რენდერის დროს). რაც ძირითადად გამოიხატება ვებ გვერდის შენელებული ჩატვირთვით .

სურ.2




სურათზე ნაჩვენებია მაგალითი, როდესაც საიტის სრულ ჩატვირთვას ჭირდება 19.29 წამი, აქედან საიტის სტილების ჩამოტვირთვას და დარენდერებას კლიენტს ბროუზერში დაჭირდა 11,79 წამი (ვებ გვერდი „გაიხსნა“ ბროუზერში ეტაპობრივად) , საიტის სტილების ჩასატვირთად განხორციელებული იყო 13 უნიკალური კავშირი სრევერთან.

Css სტილების ოპტიმიზაციისთვის საჭიროა :

1. რამოდენიმე CSS ფაილების კომბინირება ერთ (max 2 ) ფაილში;
2. საიტის იერსახის ჩასატვირთად გამოყენებული უნდა იყოს გამოყოფილი CSS ფაილი სადაც აღწერილ იქნება საიტის რენდერისთვის აუცილებელი მინიმალური პარამეტრები. მეორე ფაილში საიტის სტილების ისეთი ელემენტები უნდა იყოს აღწერილი, რაც საიტის ჩატვირთვისთვის აუცილებლობას არ წარმოადგენს;
3. დაუშვებელია CSS ფაილიდან დამატებით სხვა CSS ფაილის ჩატვირთვა ( **CSS @import** );
4. მცირე ზომის, არაქეშირებადი CSS გაწრილი უნდა იყოს HTML კოდში და არ ცალკე ფაილად. (საუბარია მაქს 10 სტრიქონიან CSS-ზე );
5. დაუშვებელია css სტილების გამოყენება `<head>` –ის გარეთ;
6. Css სტილების ჩატვირთვის დროს გასათვალისწინებელია პოზიცია (რომელი სტილი ჩაიტვირთება პირველად და რომელი მის შემდეგ);

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი საიტის შექმნის ან მოდიფიცირების დროს.

## 5.4. სურათების კომბინირება

რამოდენიმე სურათის გაერთიანება ერთ სურათში CSS SPRITE ის გამოყენებით, 50 % –ით ააჩქარებს ვებ გვერდის სურათის ჩატვირთვის ინტერვალს, შესაძლებელია საიტის ხატულების, ნიშნების, მენიუების და სახვადახვა „თემის სურათების“ კომბინირება ერთ ფაილში:

ცნობილი მენიუ apple.com საიტიდან სინამდვილეში არის 1 სურათი, ფერთა გარდასვლა და დაჭერილი ღილაკის ეფექტი ხორციელდება css sprite –ს მეშვეობით:

სურ.3


საძიებო სისტემა google –იც იყენებს ამგვარ ტექნიკას საიტის სწრაფად ჩასატვირთად და ვებ ტრაფიკის შესამცირებლად:

სურ.4


თუ ბოლო სურათს კარგად დავაკვირდებით მასში კომბინირებულია 74 სურათი, იმ შემთხვევაში თუ ეს სურათები იქნებოდა დამოუკიდებელი ფაილები მაშინ მათი რიცხვი იქნებოდა 74 და გვერდის ჩასატვირთად კლიენტსა და სერვერს შორის საჭირო გახდებოდა 74 დამოუკიდებელი TCP კავშირი, ამ შემთხვევაში სურათების კომბინირება დრამატულად ცვლის საიტის წარმადობას. იმის გამო, რომ არ მოხდეს სერვერზე რესურსების არაეფექტური გამოყენება, ასევე კლიენტის ბროუზერში საიტის ჩატვირთვის და დარენდერების დროის შემცირება გამოყებეული უნდა იყოს css sprite ტექნოლოგია.

1. Css sprite ში კომბიბნირებული უნდა იყოს სტატიკური სურათები, მენიუები, მენიუს ეფექტები და საიტის სტატიკური სტილი, სადაც გამოყენებული იქნება რაიმე ტიპის სურათი, ასევე საიტის ხატულები, და ნებისმიერი ღილაკი;

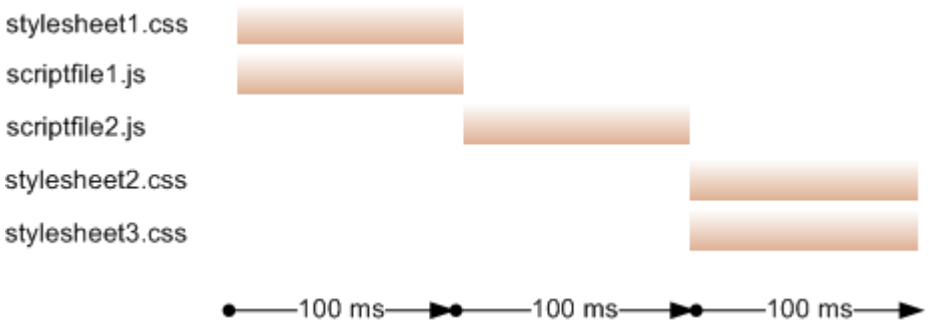
2. Css sprite –ში გამოყენებული უნდა იყოს მხოლოდ PNG ფორმატის სურათი;
3. სურათებში მასიმალურად შემცირებული ინდა იყვეს „ცარიელი ადგილი“ სურათის ხატულებს შორის;
4. Css sprite – გაერთიანებულ სურათში არ უნდა იყოს გამოყენებული 256 ფერზე მეტი, და sprite სურათი უნდა იყოს 8 ბიტიანი.

> ⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი საიტის შექმნის ან მოდიფიცირების დროს.

## 6. გვერდის სტილების და სკრიპტების ოპტიმიზაცია

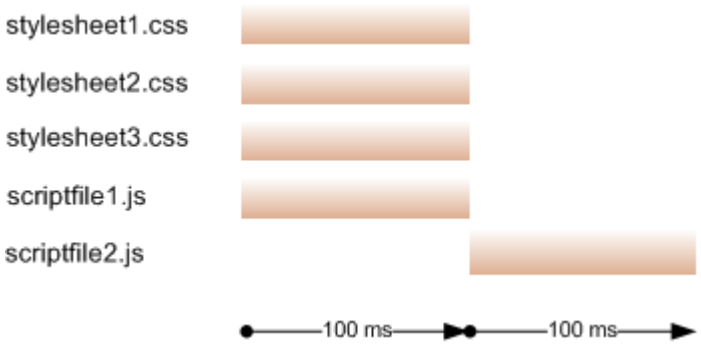
იმის გამო რომ javaScript სცენარი უნდა შესრულდეს ბოლომდე და შემდეგ გადავიდეს HTML კოდში აღწერილ სხვა სკრიპტსა თუ სტილის ჩატვირთვაზე, გასათვალისწინებელია ერთი მომენტი, თუ <head></head> სექციაში პირველად სრულდება javaScript სცენარი და შემდეგ საიტის სხვა სტილები, საიტი ჩაიტვირთება უფრო ნელა ვიდრე, იმ შემთხვევაში როდესაც საიტის <head></head> სექციაში აღწრილი იქნება ჯერ სტილები, შემდეგ javaScript სცენარი, რადგანაც css სტილები გვაძლევს ფაილის პარალელური გადმოწერის საშუალებას. (საჭირო არ არის ლოდინი საიტის კოდის გასაშვებად.) მაგ:

სურ.5



სურათზე ნაჩვენებია, რომ ჯერ იტვირთება css და javascript ფაილები პარალელურად შემდეგ, ბროუზერი ელოდება javascript შესრულებას, ამის შემდეგ გადმოიწერება და შერულდება მესამე javascript –ი, ხოლო მესამის შესრულების შემდეგ პარალელურად გადმოიწერება დარჩენილი 2 css სტილი. ჯამში ფაილების არასწორი აღწერის გამო 100 ms ით გაიზარდა გვერდის ჩატვირთვა. განვიხილოთ სწორად დაგეგმილი ფაილების განლაგება <head> სექციაში:

სურ.6




ამ შემთხვევაში სკრიპტების, სტილების ჩატვირთვას და დარენდერაბას ბროუზერი მოუნდა 200ms

*(ამ მაგალითში მოყვანილია სკიპტების უმნიშვნელო რაოდენობა, თუ გავითვალისწინებთ სკრიპტებით და სტილებით დატვირთულ საიტის მაშინ ეს დაყოვნება კატასტროფულად იმოქმედებს გვერდის ჩატვირთვაზე).*

1. პირველ რიგში უნდა ჩაიტვირთოს გვერდის სტილები სკრიპტებამდე;
2. გვერდის შიდა სკრიპტები (inline) უნდა ჩაიტვირთოს ობიექტების და რესურსების ჩატვირთვის შემდეგ;

## 6.1 JavaScript ოპტიმიზაცია

**document.write**

document.write() ფუნქციის გამოყენება გარე javascript ფაილში მნიშვნელოვნად ზრდის გვერდის ჩატვირთვის დროს, ამ დროს ბროუზერი ელოდება კიდევ დამატებით სკრიპტს და არ იწყებს გვერდის რენდერს სანამ ეს ფაილი არ ჩაიტვირთება. მაგ როდესაც first.js ფაილში წერია შემდეგი:

`document.write('<script src="second.js"><\/script>');` ბროუზერი დაელოდება სანამ არ შესრულდება first.js და second.js და ამის შემდეგ დაიწყება გვერდის დანარჩენი ელემენტების ჩატვირთვა. ანალოგიური შემთხვევის თავიდან ასაცილებლად:

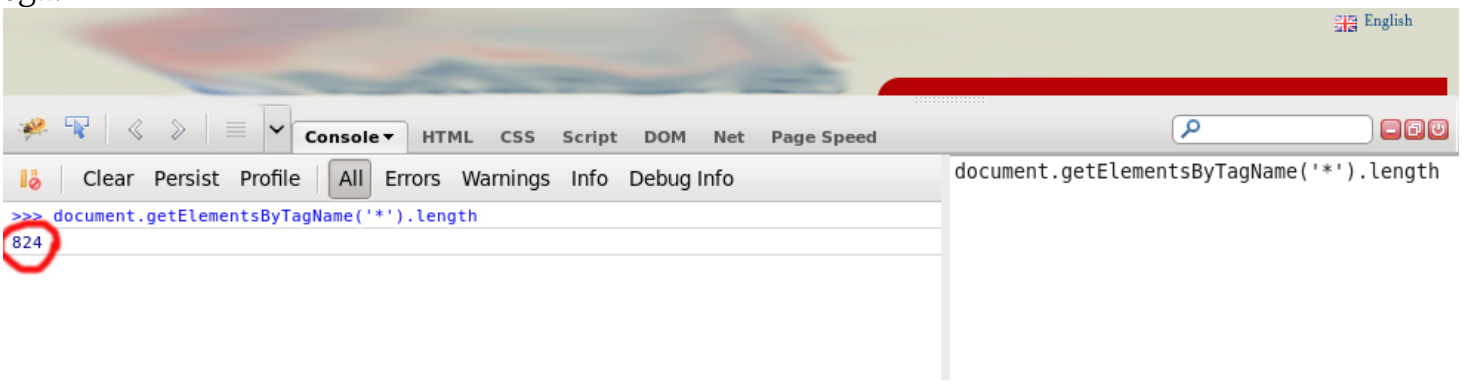
1. უნდა მოხდეს სკრიპტის დეკლარირება HTML კოდში:

```
<html>
<body>
<script src="example.js"></script>
</body>
</html>
```

## 6.2 DOM ელემენტები

DOM ელემენტების შემცირება., რაც უფრო ბევრი DOM ელემენტია სკრიპტში – უფრო მეტი აპარატურული რესურსებია საჭირო ბროზერისთის გვერდის დასარენდერებლად.

სურ.7

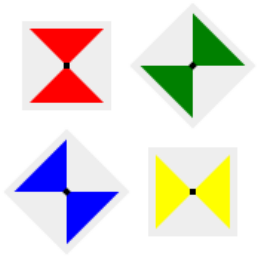



2. DOM ელემენტების რაოდენობა არ უნდა აღემატებოდეს 600–ს საიტის ყველზე დატვირთულ გვერდზე.

## 6.3 JavasSript შეკვეცა

როდესაც სკრიპტი შეიცავს არასაჭირო სიმბოლოებს ან კომენტარებს ბროუზერს ჭირდება უფრო მეტი დრო რენდერისთვის, არსებობს რამოდენიმე ცნობილი უტილიტა, რომლის მეშვეობითაც შესაძლებელია „მანქანური“: javascript–მიღება,, რომელიც რთული წასაკითხია ადმინისათვის, მაგრამ ადვილი ბროუზერისთვის, მათ ეწოდება კოდის ოპტიმიზატორები, მათი მეშვეობით შესაძლებელია მათემატიკური გათვლების გამარტივება , განვიხილოთ მაგალითი როდესაც javascript რენდერის დროს ბროუზერი ხატავს შემდეგ გეომეტრიულ იფიგურებს:

სურ.7



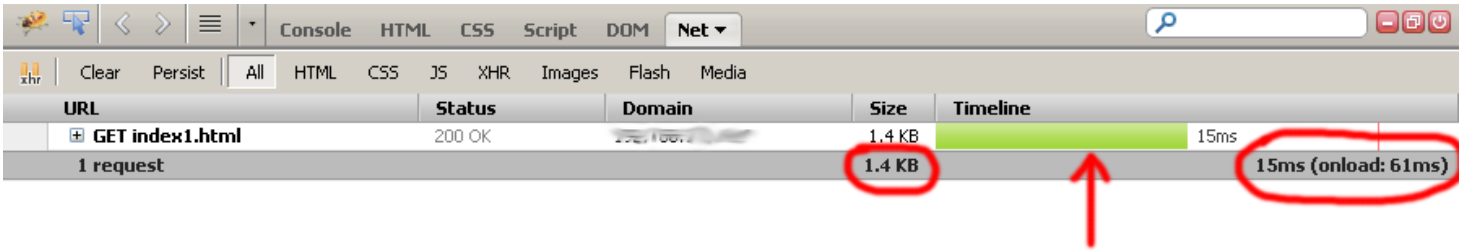
ასეთი სურათის მისაღებად დაწერილი იყო შემდეგი კოდი:

```
<html>
  <head>
   <script type="application/javascript">
 function drawBowtie(ctx, fillStyle) {   ctx.fillStyle = "rgba(200,200,200,0.3)";
ctx.fillRect(-30, -30, 60, 60); ctx.fillStyle = fillStyle;ctx.globalAlpha = 1.0;
ctx.beginPath(); ctx.moveTo(25, 25); ctx.lineTo(-25, -25); ctx.lineTo(25, -25);
ctx.lineTo(-25, 25); ctx.closePath(); ctx.fill(); }  function dot(ctx) {
ctx.save();ctx.fillStyle = "black"; ctx.fillRect(-2, -2, 4, 4);ctx.restore();}
function draw() {var canvas = document.getElementById("canvas");var ctx =
canvas.getContext("2d");/* note that all other translates are relative to this *//*
one */  ctx.translate(45, 45); ctx.save(); /*ctx.translate(0, 0);  unnecessary */
drawBowtie(ctx, "red"); dot(ctx); ctx.restore(); ctx.save(); ctx.translate(85, 0);
ctx.rotate(45 * Math.PI / 180); drawBowtie(ctx, "green"); dot(ctx);ctx.restore();
ctx.save(); ctx.translate(0, 85); ctx.rotate(135 * Math.PI / 180);drawBowtie(ctx,
"blue");dot(ctx);ctx.restore(); ctx.save(); ctx.translate(85, 85);ctx.rotate(90 *
Math.PI / 180); drawBowtie(ctx, "yellow"); dot(ctx); ctx.restore(); }
</script>
  </head>
  <body onload="draw()"><canvas id="canvas" width="300" height="300"></canvas>
  </body>
```

```
</html>
```

აღნიშნული სცენარი განთავსებულია index1.html ფაილში, მის ჩასატვირთად საჭიროა 15ms და რენდერისთვის 61ms

სურ.8



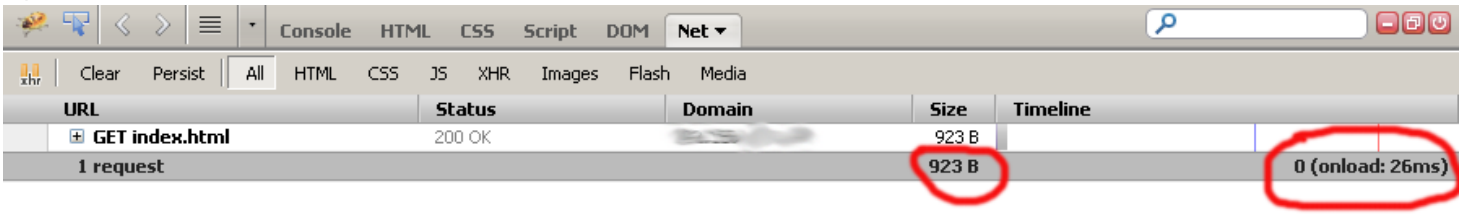

თუ მოვახდენთ აღნიშნულ javaScript კოდის „მინიფიცირებას“ **YUI Compressor** უტილიტით და გავხდით მას რენდერისთვის ოპტიმიზირებულს, მივითღებთ შემდეგ კოდს:

```
<html>
  <head>
<script type="application/javascript">
 function drawBowtie(a,b){a.fillStyle="rgba(200,200,200,0.3)";a.fillRect(-30,-
30,60,60);a.fillStyle=b;a.globalAlpha=1;a.beginPath();a.moveTo(25,25);a.lineTo(-25,-
25);a.lineTo(25,-25);a.lineTo(-25,25);a.closePath();a.fill()}function
dot(a){a.save();a.fillStyle="black";a.fillRect(-2,-2,4,4);a.restore()}function
draw(){var b=document.getElementById("canvas");var
a=b.getContext("2d");a.translate(45,45);a.save();drawBowtie(a,"red");dot(a);a.restor
e();a.save();a.translate(85,0);a.rotate(45*Math.PI/180);drawBowtie(a,"green");dot(a)
;a.restore();a.save();a.translate(0,85);a.rotate(135*Math.PI/180);drawBowtie(a,"blue
");dot(a);a.restore();a.save();a.translate(85,85);a.rotate(90*Math.PI/180);drawBowti
e(a,"yellow");dot(a);a.restore()};
 </script>
  </head>
  <body onload="draw()"><canvas id="canvas" width="300" height="300"></canvas>
  </body>
 </html>
```

მინიფიცირებული **javaScript** კოდი ჩასმულია index.html ფაილში, აღნიშნული გვერდის ერთდროული ჩასატვირთვის და რენდერისთვის ბროუზერს დაჭირდა 26მს, (1მს –ზე ნაკლები გვერდის ჩასატვირთად, გვერდის მოც=923 ბაიტი და სცენარის რენდერისთვის 26ms ) წინა

მაგალითისგან განსხვავებით ამ შემთხვევაში მივიღეთ 43.3% იანი წარმადობა გვერდის ცატვირთვის დროს, მხოლოდ ერთი ფაილის ოპტიმიზაციით.

სურ.9




3. javaScript მინიფიკაციისთვის საჭირიოა სკრიპტის ოპტიმიზაცია განხორციელდეს შემდეგი უტილიტებით: **Closure Compiler, YUI Compressor, JSmin**

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი საიტის შექმნის ან მოდიფიცირების დროს.

## 7. ვებ გვერდის შეკუმშვა

გვერდის შეკუმშვავში იგულისხმება HTML, CSS, Javascript შეკუმშვა Gzip ან deflate ალგორითმების მეშვეობით, შეკუმშვის აზრი მგომარეობს შემდეგში, სერვერი კუმშავს კონტენტს, უგზავნის მას კლიენტს და კლიენტის ბროუზერი იწყებს შეკუმშული კონტენტის განკუმშვას, შემდეგ მის რენდერს, შესაბამისად კლებულობს ქსელში გადაცემადი ინფორმაციის მოცულობა, განვიხილოთ მაგ:

| ფაილი | მოცულობა კბ–ში | მოცულობა კომპრესიის შემდეგ (gzip) | კომპრესია პროცენტებში |
|---|---|---|---|
| main.js | 92 kb | 20kb | 21.73% |
| theme.css | 64kb | 12kb | 18.75% |
| page.html | 76kb | 16kb | 21.05 |
| image.png | 150kb | 150kb | 0% |
| ჯამი: | 382kb | 198kb | 61.53% |

აღნიშნულ მაგალითში შეიკუმშა რესურსების 61.53 %  და შესაბამისად მის გადაწერას ქსელში მოუნდება 61.53% –ით უფრო ნაკლები დრო ვიდრე შეუკუმშავი ფაილების დროს.

თანამედროვე ბროუზერების უმეტესობა გებულობს შეკუმშულ კონტენტს და ამუშავებს მათ, (IE-6 ის გარდა, აგრეთვე ზოგიერთი ანტი–ვირუსული პროგრამა, ცვლის http-header'ს და შესაბამისად ბროუზერამდე იგზავნება  შეცვლილი შიგთავსი).

1. შეკუმშვის დროს გამოყენებუილი უნდა იყოს gzip ალგორითმი;
2. არ შეიძლება ბინარული ფაილების შეკუმშვა, რადგანაც ისინი უკვე არიანა შეკუმშულები თავისი კომპრესიის ალგორითმებით (მაგ: *.jpg, *.pdf, *.swf…);
3. შესაკუმში კონტენტის მოცულობა უნდა მერყეობდეს 150 დან 1000 ბაიტს შორის;

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი საიტის შექმნის ან მოდიფიცირების დროს.

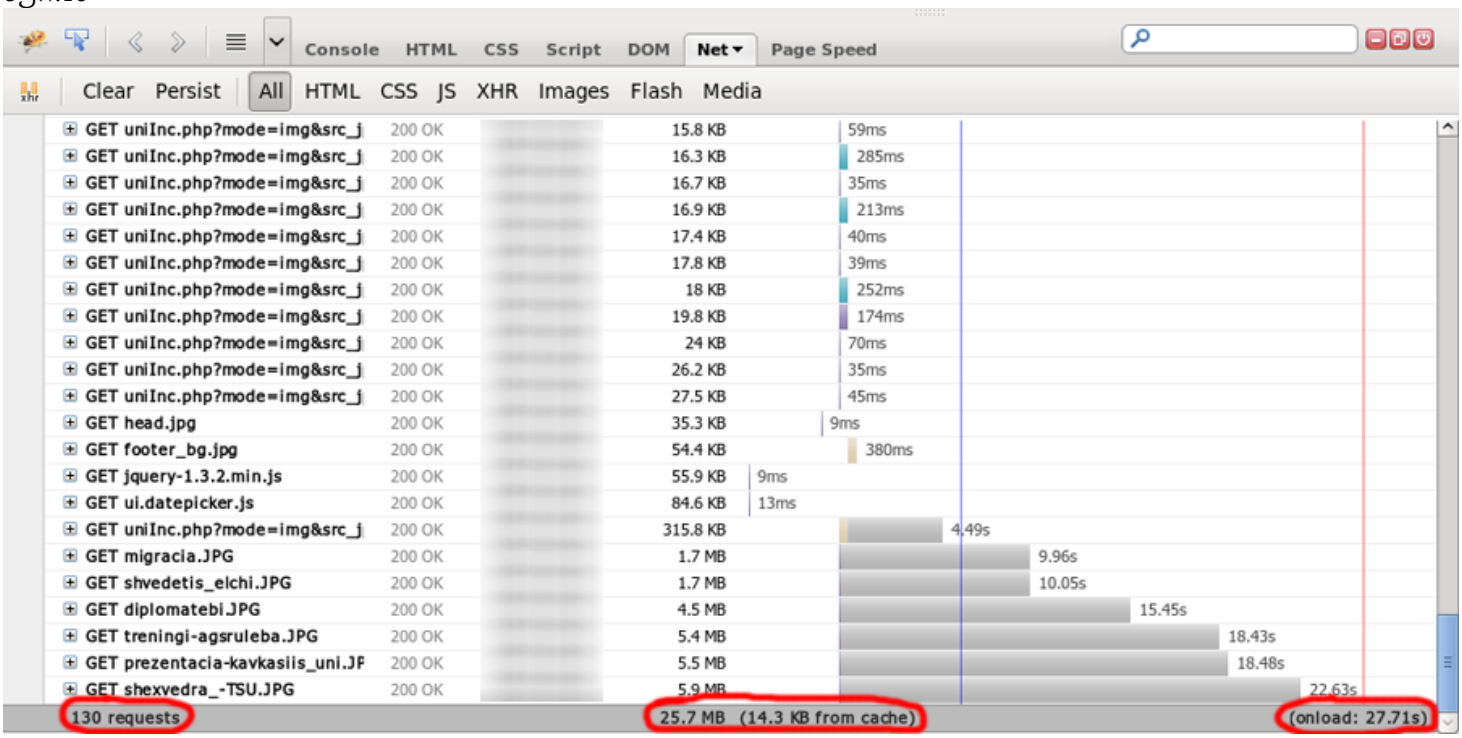
⚠ საიტის ადმინისტრატორი ვალდებულია უზრუნველყოს კომპრესიის ალგორითმების გამართული მუშაობა ვე ბ სერვერზე.

# 8.სურათების ოპტიმიზაცია

კარგად ოპტიმიზირებული და შეკუმშული სურათები შეამცირებს ქსელის ტრაფიკს. სხავდასხვა კომპიუტერულ პროგრამებში დამუშავებულ სურათებს შეიძლება ქონდეთ არასაჭირო კომენტარების ათეულობით სტრიქონი, ან არასწორი ფერთ გამა, რაც ზრდის ფაილის მოცულობას და აძნელებს მის რენდერინგს ბროუზერში, ასევე გასათვალისწინებელია სურათის გარჩევადობა xy. ხშირია შემთხვევები, როდესაც არ ხდება სურთის ზომების სწორი შერჩევა და არც მისი დამუშავება და შესაბამისად ვებ გვერდზე იტვირთება სურათი სრული გარჩევადობით, ამ დროს სურათი კლიეტის ბროუზერში იტვირთება საწყისი ზომით მაგარმ html scaling პარამეტრის გამოყენებით ჩანს როგორც პატრა ზომის, მაგ: გვაქ სურთი, რომლის მოცულობა არის 6.6mb და გარჩევადობაა 5184x3456 პიქსელი, თუ HTML კოდში ამ სურათს განვუსაზღვრავთ სიმაღლეს და სიგანეს, მოხდება scaling: `<img width="170" height="113" src="files/Photos/heavy.JPG" >` სინამდვილეში სურათი სრული ზომით და მოცულობით ჩაიტვირთება, წარმოვიდგინოთ შემთხვევა, როდესაც ასეთი სურათები რამოდენიმეა ვებ გვერდზე, მაშინ კლიენტის ბროუზერში ჩატვირთული ვებ გვერდის ზომამ შეიძლება გადააჭარბოს 25mb-s, სურათის ამგვარი ჩასმა HTML კოდში არის დიდი შეცდომა, რადგანაც ამის გამო შეიძლება:

1. დაიხარჯოს უამრავი ქსელური რესურსი;
2. შემცირდეს სერვერის გამტარუნარიანობა (ერთი გვერდის ჩატვირთვა = 25 ბ download, თუ სიატს დღეში 200 ვიზიტორი სტუმრობს და საშუალოდ თითო ვიზიტორი საიტზე 10 გვერდს ათვალიერებს გამოდის, რომ სერვერიდან ყოველდღიურად მხოლოდ გვერდების მონახულების პროცესში იტვირთება **50GB** ინფორმაცია, რაც წარმოუდგენალად დიდია ინფორმაციული ტიპის საიტებისთვის. )
3. სერვერული აპარატურის არაეფექტური გამოყენება, (საჭირო ხდება დამატებითი გამოთვლითი ოპრაციების ჩატარება, რაც აპარატურის გაძლიერებასთანაა დაკავშირებული)
4. მომხმარებლის ბროუზერის და მთლიანი კომპიუტერის „გაჭედვა“ (ბროუზერს უწევს დაარენდერსოს 25 მბ მხოლოდ მედია ობიექტები და ამასთანავე იტვირთება სტილები და სკრიპტები, რომლებიც მოქმედებენ აღნიშნულ ობიექტებზე, იწვევევენ სურათების „გადახატვას“ (repaint) - jquery და სხვა.);
5. ჰოსტინგ ინფრასტრუქტურის შენელება, და ახალი http მოთხოვნების მოლოდინში ჩაყენება – TIME_WAIT;

სურ.10




**სურათზე ნაჩვენებია ვებ გვერდი, რომლის ჩატვირთვის დროს ბროუზერში იტვირთება 25.7 მბ ინფორმაცია, სერვერთან კავშირების რაოდენობა აღწევს 130–ი და სრული გვერდი ჩატვირთვას ჭირდება 27,71 წამი.**

**აღნიშნული პრობლემის მოსაგვარებლად აუცილებელია:**

1. სურათების ავტომატური დაპატარავება, thumbnail ფორამატში,
2. აღნიშნულ ფორმატში სურათის მაქსიმალური მოცულობა არ უნდა აღემატებოდეს : 30 კბ–ს,
3. Thumbnail სურათის სტანდარტული გარჩევადობა უნდა იყოს 175x125 პიქსელი.
4. უნდა მოხდეს ორიგინალი ფაილების დამუშავება, მათი მაქს ზომა არ უნდა აღემატებოდეს 160 კბ–ს;
5. ორიგინალი ფალების დამუშავების შემდეგ მათი მქს გაფართოება არ უნდა აღემატებოდეს 900x700 პიქსელს.
6. დასაშვები **ფოტო** სურათის ფორმატია მხოლოდ: JPG*
7. უნდა მოხდეს სურათების ოპტიმიზაცია სხვადასხვა უტილიტებით სერვრზე( **jpegtran** an **jpegoptim**).

8. მომხმარებლის მიერ სურთების ატვირთვის დროს განსაზღვრული უნდა იყოს სურათის მაქს დასაშვები ასტვირთი მოც: 5მბ და სურათი ფორმატი:JPG, სხვა ტიპის და მოცულობის სურათების ატვირთვა დაუშვებელია;
9. სურათის ზომების და მოცულობების შესაცვლელად გამოყენებული უნდა იყოს, მხოლოდ php-gd2 ბიბლიოთეკა.
10. ორიგინალი (დიდი ზომის) სურათები არ უნდა იყოს ჩატვირთული საიტის ჩატვირთვასთან ერთად, მათი ჩატვირთვა უნდა მოხდეს, როდესაც მომხმარებელი დაკლიკავს სურათზე და არა წინასწარ.
11. სურათები არ უნდა იყოს CMYK რეჟიმში და არ უნდა იყოს 8 ბიტზე მეტი.
12. სერვერზე არ უნდა ინახებოდეს 900x700 მეტი ზომებისი მქონე სურათები და მათი მოცულობა არ უნდა აღემატებოდეს 160კბს.

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი 7-ე ს გარდა საიტის შექმნის ან მოდიფიცირების დროს.

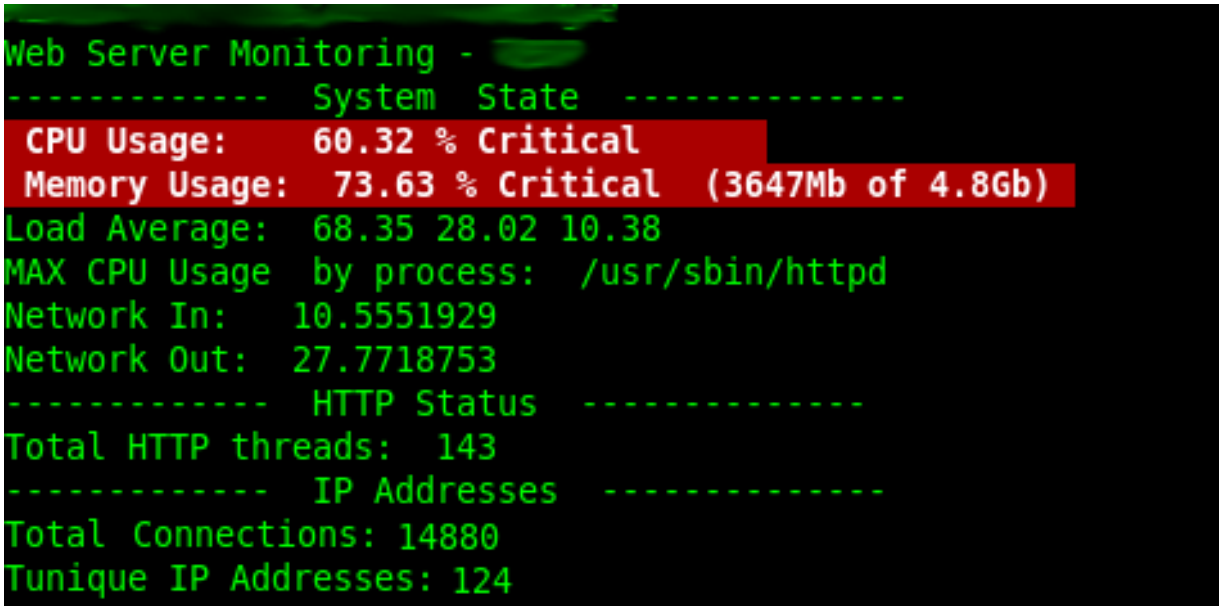
⚠ საიტის ადმინისტრატორი ვალდებულია უზრუნველყოს 7-ე პუნქტის განხორციელება ვებ სერევერზ ე.

## 9. კავშირების რაოდენობა

როდესაც მომხმარებელი ხსნის ვებ გვერდს ამ გვერდის ჩატვირთვასთან ერთად იტვირთება HTML კოდი, CSS სტილები, JavaScript სკრიპტები, სურათები, ფლეშ ანიამაციები და სხვა აუდიო/ვიდეო მედია, ამ დროს სერვერსა და კლინტს შორის იზრდება კავშირების რაოდენობა თითოეულ ჩამოტვირთულ ობიექტზე.

**1 ფაილი = 1 კავშირი,** რაც უფრო ბევრია კავშირების რაოდენობა სერვერთან უფრო დიდი დრო ჭირდება ვებ გვერდის გახსნას და შესაბამისად უფრო მეტი აპარატურული რესურსი ხდება საჭირო სრევრის მხარეს, რათა გაუმკლავდეს კავშირების მასიურ ზრდას. ვებ საიტის ყველაზე მთავარ როლს მისი ოპტიმიზირების კუთხით თამაშობს- HTTP კავშირების რაოდენობა. კავშირების ზრდასთან ერთად იზრდება round-trip (RTT).

რადგანაც ბროუზერს შეუძლია გადმოქაჩოს მხოლოდ 2 კომპონენტი პარალელეურად, თითო ჰოსტიდან და ყოველ http კავშირს გააჩნია round-trip დაყონება 0,2 წამი, გამოდის რომ 20 http request–ის დროს დაყოვნება იქნება 2 წამი, თუ გავითვალისწინებთ, რომ ბროუზერები ხარჯავენ თავისი დროის 80% კოდის სინტაქსის "პარსვაში", სტილების "ხატვაში", სკრიპტის სცენარის შესრულებაში და მედია ობიექტების შესრულებაში, გამოდის რომ თუ შევამცირებთ კავშირების რაოდენობას გავზრდით ვებ გვერდის წარმადობას, და შევამციცერბთ სერვერის გადატვირთვას.
სურ.11




სურათზე ნაჩვენებია: გვერდის კავშირების რაოდენობის ზრდის გამო სერვერის რესურსების არაეფექტური გამოყენება, 124 უნიკალური ვიზიტორი, რომლებიც ერთდროულად სტუმრობენ ვებ გვრედს. ვებ გვერდის ჩასატვირტად საჭიროა 120 HTTP request, შესაბამისად კავშირების რაოდენობა აღწევს 14880 რასაც ჭირდება

3647mb ოპერატიული მეხსიერება და 4 ბირთვიანი 2.8 ghz -იანი პროცესორის 60.32 %, რაც ძალიან ბევრია იმისთვის რომ სერვერმა მოემსახუროს 124 ვიზიტორს ერთდროულად. (სურათზე ვიზიტორების კავშირების რაოდენობა – „**Tunique IP Address**: “ გამოსახულია მხოლოდ ESTABLISHED მდგომარეობაში, ხოლო „**Total Connections**:“ დაჯამებულია
`TIME_WAIT+ESTABLISHED+SYN_RECV+FIN_WAIT2+FIN_WAIT1+CLOSING+CLOSE_WAIT`
მგომარეობებით),

ანალოგიური სიტუაციების თავიდან ასაცილებლად საჭიროა:

1. კავშირების რაოდენობის შეზღუდვა;
2. ყველზე გადატვირთული საიტის გვერდის კავშირების რაოდენობა სერვერან არ უნდა აღემატებოდეს 60–ს სურათებიანად;
3. გვერდის კავშირები, ფოტო სურათების გარეშე არ უნდა იყოს 40 –ზე მეტი;

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი საიტის შექმნის ან მოდიფიცირების დროს.

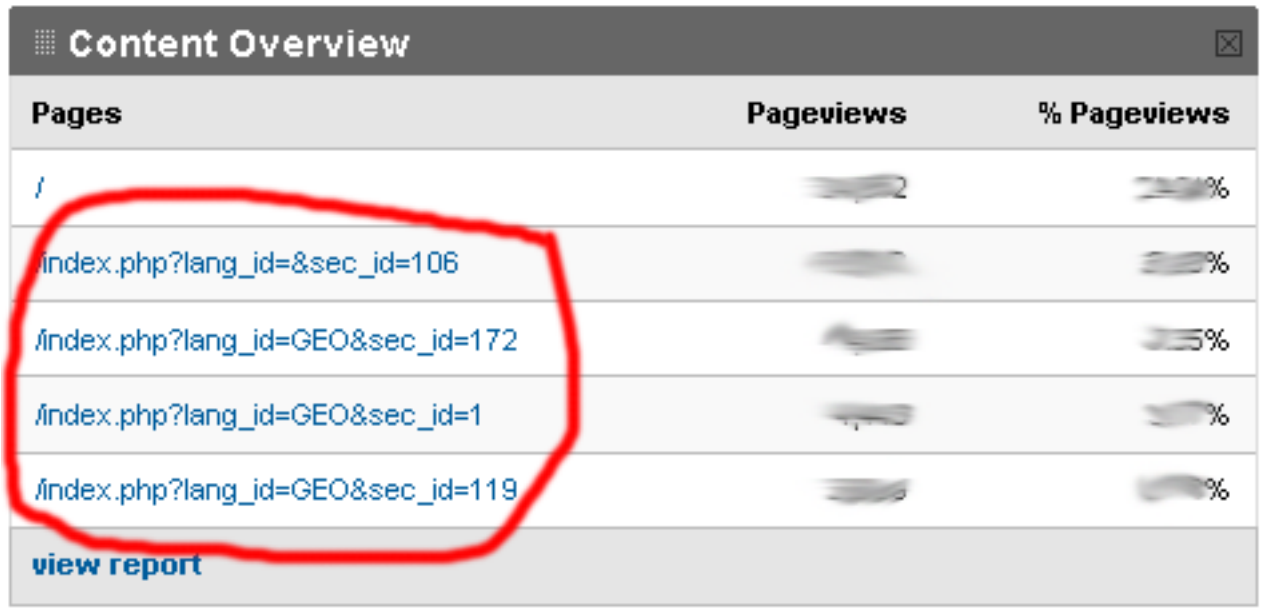
# 10. წმინდა ბმულები

ჩვეულებრივ ბმულებში იგულისხმება ვებ მისამართის ბმული, რესურსზე, მაგ. ჩვეულებრივ ბმულში:
http://www.site.ge/index.php?lang_id=GEO&sec_id=23&mod_id=0&info_id=0&new_year=0&limit=0&date=&new_month=&entrant=4

წერია ადმიანისთვის ძნელად გასაგები სიტყვების და ციფრების კომბინაცია, საიტის სხვადასხვა ცვლადები და POST მოთხოვნები, ასეთ ბმულების გამოყენებას გააჩნია რამოდენიმე ცუდი მხარე:

1. ძნელად  დასამახსოვრებელია მომხმარებლისთვის ;
2. არის დიდი ალბათობა  ბმულში მოყვანილი ცვლადების გადასინჯვის დროს აღმოჩენილმა შეცდომამ გამოიწვიოს SQL Injection ან სხვა ტიპის შეცდომა;
3. არასწორად ინდექსირედება საიტის კატეგორია, ან რესურსი საძიებო სისტემაში;
4. საიტის ანალიტიკის ნახვის დროს შეუძლებელია გაიგო, თუ  რომელ გვრედს სტუმრობდა მომხამარებელი, საიტის ბმულსზე  გადაუსვლელად;

სურ.12




შესაძლებელია ბმულების „გასუფთავება“ „Clean URL“. ასეთ შემთხვევაში ბმულები ხდება მარტივად კითხვადი და ადვილად დამახსოვრებადი, მაგ თუ სიატის  გვერდი არის:
http://www.site.ge/index.php?lang_id=&sec_id=106
წმინდა ბმულის ჩართვის შედეგად კი იქნება:
http://www.site.ge/contact

1. წმინდა ბმულების ჩასართავად გამოყენებული ინდა იყოს .httacces დირექტივების ფაილი;
2. ვებ საიტების ბმულები განსაზღვრული  უნდა იყოს რესურსების სახელების მიხედვით;

3. თითოეული ახალი გვერდი უნდა თარიღდებიოდეს მიმდინარე თარიღით: /news/yyy/mm/dd (http://site.ge/news/2011/03/15)

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი საიტის შექმნის ან მოდიფიცირების დროს.

# 11. PHP ოპტიმიზაცია

დიმანიური ვებ გვერდის ჩატვირთვის დროს გვერდი ჯერ სრულდება სერვერზე და შემდეგ ეგზავნება კლიენტს შერულებული html კოდის სახით, ვებ გვერდის შესრულებას საშულოდ ჭირდება 500ms სერვერზე, ამ დროს კლიენტის ბროუზერი ელოდება სანამ სერვერი გამოუგზავნის შერულებულ კოდს, php შემთხვევაში შესაძლებელია **flush()** გამოყენება, და შერულებული კოდის ეტაპობრივი გამოგზავნა კლიენტთან , php flush() -ის გამოყენება ოპტიმალურია თუ ის არის <head>და <body> სექციებს შორის მაგ:

```
  ... <!-- css, js -->
</head>
<?php flush(); ?>
<body>
  ... <!-- content -->
```

ამ დროს კლიენტს ეგზავნება სტულები და სკრიპტები და სანამ გვერდი სრულიად გამოეგზავნება ბროუზერს შეუძლია ამ შუალედში დაარენდეროს და შეასრულოს საიტის კოდი.

**Sendfile** – ეს პარამეტრი მნიშვნელოვნად აჩქარებს სტატიკური კონტენტის გადაცემას კლიენტის ბროუზერამდე, ზედმეტი ჩაწერა წაკითხვის თავიდან აცილების გზით სერვერზე, მაგრამ shared storage (საზიარო სივრცის) დროს როდესაც რამოდენიმე HTTP სერვერი დამოკიდებულია ერთ საზიარო სივრცეზე აუცილებელია ამ პარამეტრის გამორთვა, რადგანაც ეს პარამეტრი მხარდაჭრილია ოპრეციული სისტემის დონეზე, shared storage –ს შემთვევაში სისტემის კერნელი ვეღარ ემსახურება ახალ მოთხოვნებს ქეშიდან .

საიტის კოდის დიზაინის დროს გათვალისწინებული უნდა იყოს:

1. Php flush() მეთოდის გამოყენება;
2. Error Handling – შეცდომების დაბეჭდვა უნდა იყოს გამორთული, შედეგად მნიშვნელოვნად მცირდება სერვერზე I/O ოპერაციების შესრულების დრო.
3. register_globals პარამეტრი უნდა იყოს გამორთული!
4. მონაცემთა ბაზასთან კავშირის დროს უნდა გადამოწმდეს ბაზის კავშირის დახურვა;

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი 2-ე ს გარდა საიტის შექმნის ან მოდიფიცირების დროს.

⚠ საიტის ადმინისტრატორი ვალდებულია უზრუნველყოს 2–ე პუნქტის ფუნქციონირება საჭიროებისამებრ.

# 12. რენდერის ოპტიმიზაცია ბროუზერში

## 12.1 სურათების ზომა

სურათების ზომების მითითება მნიშვნელოვანად აჩქარებს გვერდის ჩატვირთვას, ბროუზერს აღარ ჭირდება სურათის „გადახატვა“ და მატემატიკური კალკულაციები. სურათებსი ჩატვირთვის დროს უნდა იყოს მითითებული სურათის ზუსტი ზომები, მაგ დაუშვებელია სურათის სიმაღლის და სიგანის მითითება 30x30 <img> ტეგში, თუ მისი რეალური ზომაა 60x60.

## 12.2 CSS სტილები <head> სექციაში

საიტის სტილები აუცილებლად უნდა იყოს განთავსებული <head> სექციაში, მათი განთავსება სხვა ნებისმიერ სექციაში მნიშნელოვნად ანელებს საიტის ჩატვირთვის წარმადობას. <style> ბლოკებში განთავსებული css წესებიც აუცილებლად უნდა იყოს განთავსებული აღნიშნულ სექციაში.

## 12.3 სიმბოლოების მითითება

**<head>** სექციაში meta ტეგი მითითებული უნდა იყოს ყველაზე ადრე, რადგანაც ბროუზერი ეძებე მას პირველი 1024 ბაიტის დარენდერების დროს, შესაბამისად მისი მითითება სექციის ქვემოთ იწვევს წარმადობის პრობლემებს.

## 12.4 კონტენტის ტიპი

აუცილებელია მეტა ტეგში კონტენტის ტიპის მითითება, რადგავნაც, თუ კონტენტის ტიპი არ არის მითითებული ბროუზერი ცდილობს გამოიცნოს საიტის კონტენტი სხვადასხვა ალგორითმების გამოყენებით, რაც ანელებს გვერდის რენდერს. ამიტომ მისათითებელია კონტენტის ტიპი აბსოლიტურად ყველა გამოყენებული რესურსისათვის:

```
text/html text/plain text/css image/jpeg image/x-png audio/basic text/javascript
```

## 12.5 სიმბოლოების კოდირება

აუცილებელია სიმბოლოების კოდირების მითითება (character encoding) meta ტეგში, იმ სოიმბოლების კოდირება, რომლებიც შედგომში გამოყენებული იქნება HTML კოდში. თუ ბროუზერმა აღმოაჩინა, რომ მთითებული კოდირება განსხვავადება კოდში გამოყენებული კოდირებისაგან, ის შეეცდება გამოიცნოს სწორი კოდირება და შემდეგ დაარენდეროს მთელი გვერდი, რაც ანელებს გვერდის ჩატვირთვის წარმადობას.

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს ყველა ჩამოთვლილი პუნქტი საიტის შექმნის ან მოდიფიცირების დროს.

# 13 ქეშის ოპტიმიზაცია

აღნიშნული საინფორმაციო უსაფთხოების ბიულეტინის ქეშის ოპტიმიზაციაში იგულისხმება: HTTP Accelerator – Revers Proxy და სხვა ინფრასტრუქტურული გადაწყვეტილებები, რომლებიც აქეშირებენ მხოლოდ სტატიკურ კონტენტს : საიტის სტილებს, სკრიპტებს და მედია ობიექტებს, შედეგად http სერვერები თავისუფლდებიან 40-50% იანი სტატიკური კონტენტის მომსახურების დატვირთვისგან.

⚠ საიტის დეველოპერი ვალდებულია გაითვალისწინოს აღნიშნული საიტის შექმნის ან მოდიფიცირების დროს.

⚠ საიტის ადმინისტრატორი ვალდებულია უზრუნველყოს ამ პარამეტრის ფუნქციონირება ვებ სერვერზე.

# 15. ტერმინოლოგია და განმარტება