



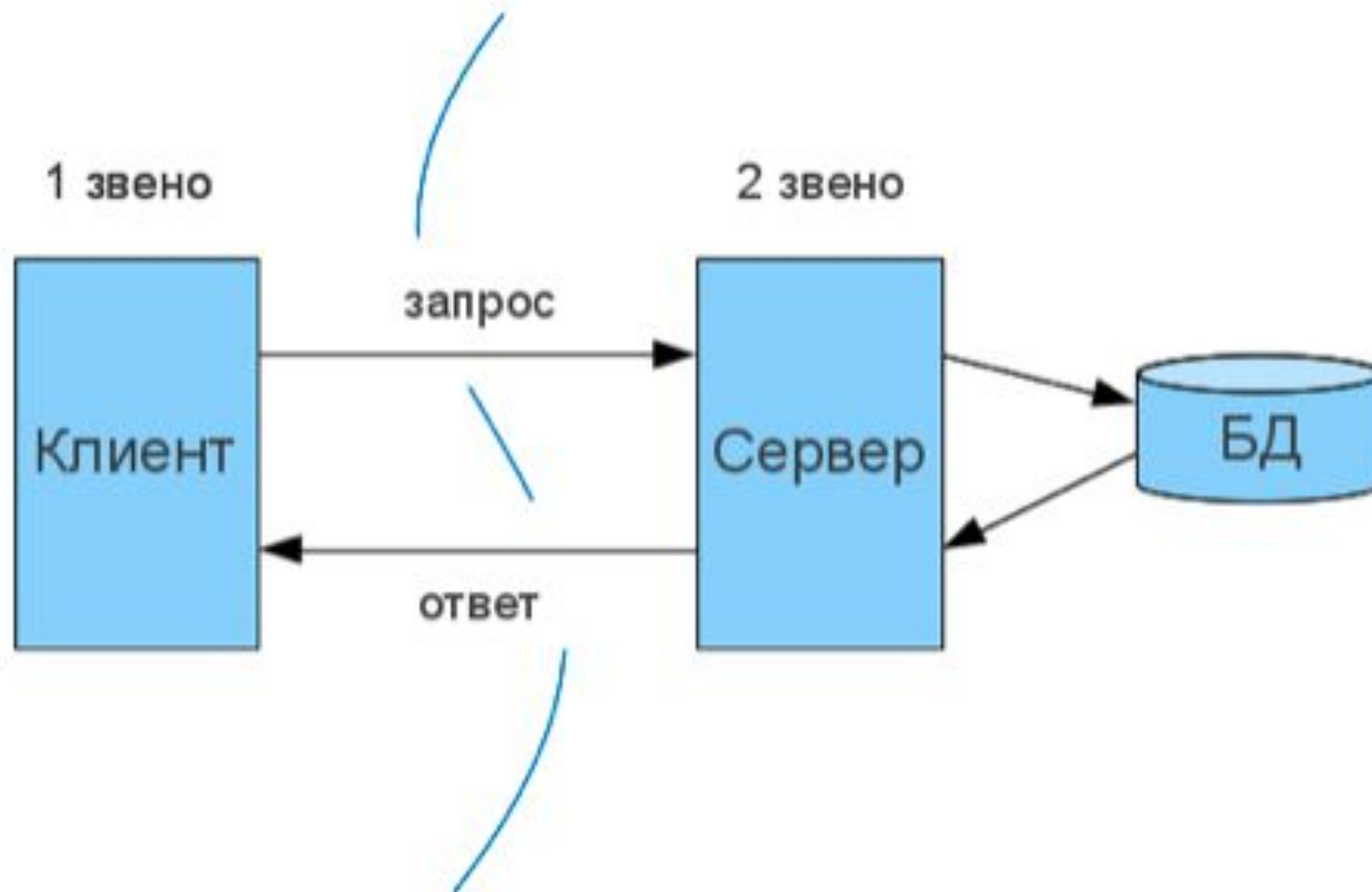
**Архитектура клиент-  
сервер.  
Тестирование web**

- Архитектура клиент-сервер.
- Клиент-серверные технологии.
- Сетевые протоколы.
- Тестирование web.
- Виды уязвимостей.
- Основные виды нагрузочного тестирования.
- Кросс-браузерное тестирование.
- Инструменты разработчика.

**Архитектура «клиент-сервер»** определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций).



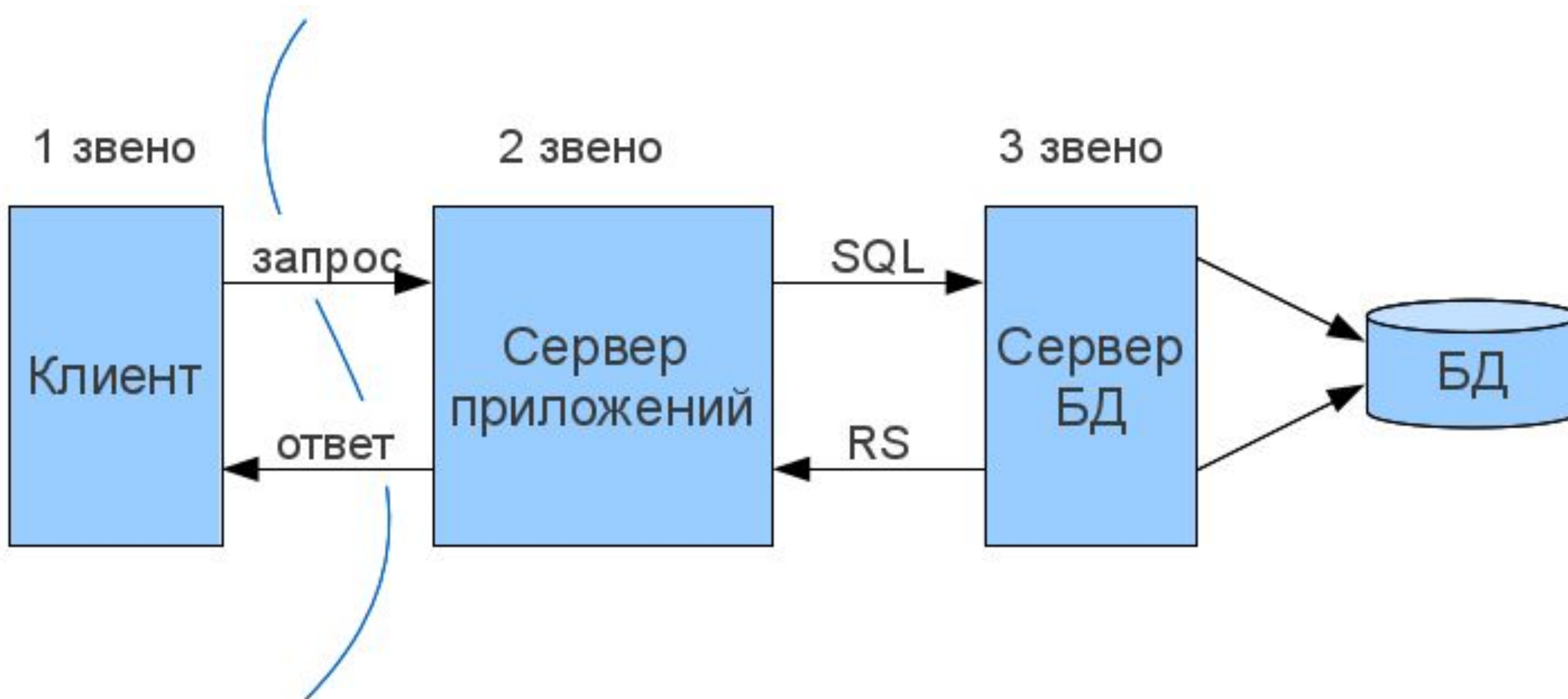
# Двухзвено



Основные модели взаимодействия в рамках двухзвенной архитектуры:

- **Сервер терминалов** — распределенное представление данных.
- **Файл-сервер** — доступ к удаленной базе данных и файловым ресурсам.
- **Сервер БД** — удаленное представление данных.
- **Сервер приложений** — удаленное приложение.

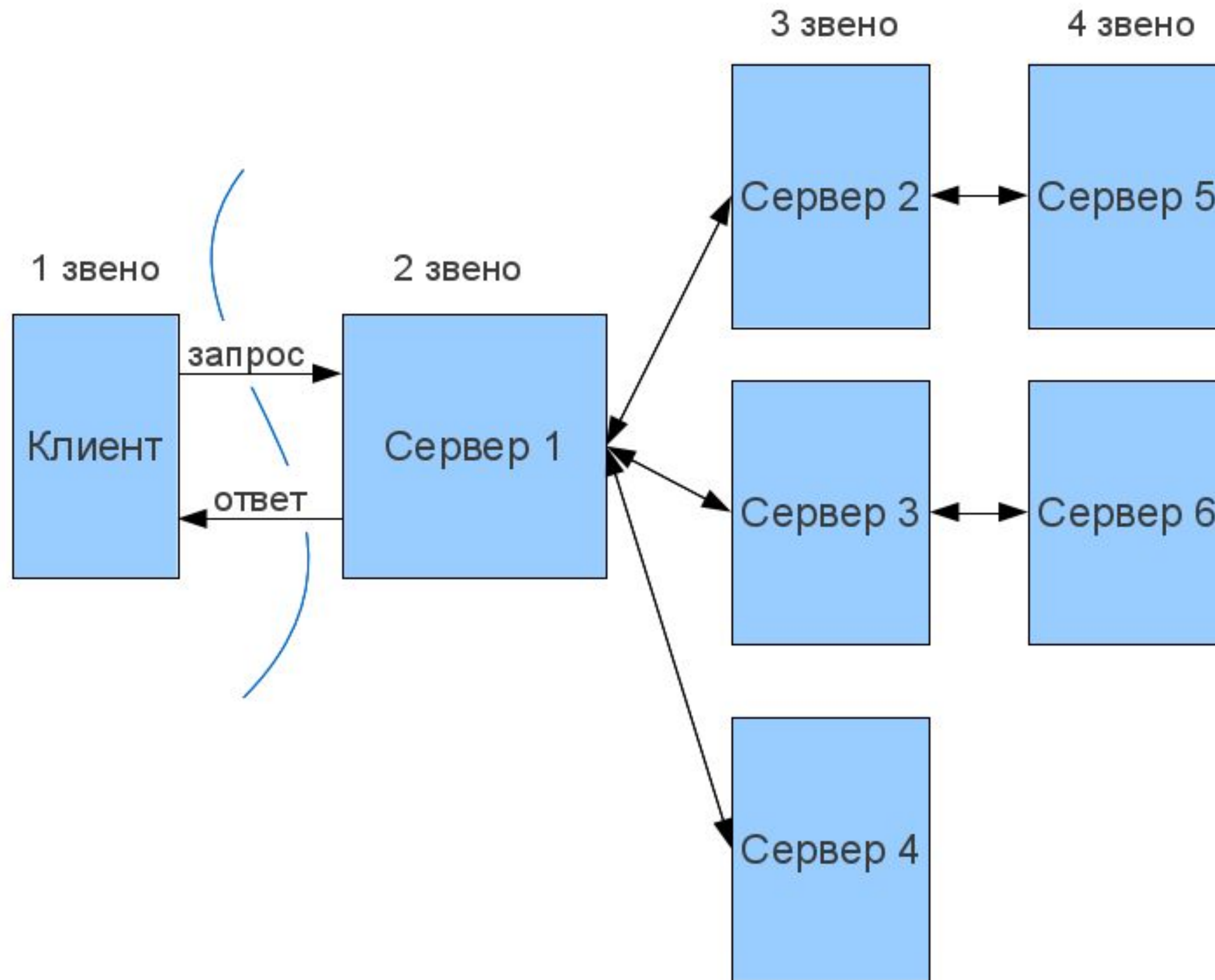
# Трехзвенная архитектура



Компоненты в трехзвенной архитектуре распределяются следующим образом:

- **Представление данных** — на стороне клиента.
- **Прикладной компонент** — на выделенном сервере приложений
- **Управление ресурсами** — на сервере БД, который и представляет запрашиваемые данные.

# Многозвено





## Типы сервисов:

- **Web-серверы.**
- **Серверы приложений.**
- **Серверы баз данных.**
- **Файл-серверы.**
- **Прокси-сервер.**
- **Файрволы (брандмауэры).**
- **Почтовые серверы.**
- **Серверы удаленного доступа (RAS)**



## **«Тонкий» клиент**

Клиент, вычислительных ресурсов которого достаточно лишь для запуска необходимого сетевого приложения через web-интерфейс. Пользовательский интерфейс такого приложения формируется средствами статического HTML (выполнение JavaScript не предусматривается), вся прикладная логика выполняется на сервере.

## **«Толстый» клиент**

Рабочая станция или персональный компьютер, работающие под управлением собственной дисковой операционной системы и имеющие необходимый набор программного обеспечения. К сетевым серверам «толстые» клиенты обращаются, в основном, за дополнительными услугами (например, доступ к web-серверу или корпоративной базе данных).

**Сетевой протокол** — набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами

TCP/IP	OSI Model	Protocols
Application Layer	Application Layer	DNS, DHCP, FTP, HTTPS, IMAP, LDAP, NTP, POP3, RTP, RTSP, SSH, SIP, SMTP, SNMP, Telnet, TFTP
	Presentation Layer	JPEG, MIDI, MPEG, PICT, TIFF
	Session Layer	NetBIOS, NFS, PAP, SCP, SQL, ZIP
Transport Layer	Transport Layer	TCP, UDP
Internet Layer	Network Layer	ICMP, IGMP, IPsec, IPv4, IPv6, IPX, RIP
Link Layer	Data Link Layer	ARP, ATM, CDP, FDDI, Frame Relay, HDLC, MPLS, PPP, STP, Token Ring
	Physical Layer	Bluetooth, Ethernet, DSL, ISDN, 802.11 Wi-Fi



## HTTP/HTTPS

- HTTP - протокол используется для передачи произвольных данных.
- HTTPS (HyperText Transfer Protocol Secure) - расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов SSL или TLS.



Http



Https

## HTTP

Всё ПО для работы с протоколом HTTP разделяется на три большие категории:

- Серверы как основные поставщики услуг хранения и обработки информации (обработка запросов);
- Клиенты - конечные потребители услуг сервера (отправка запроса);
- Прокси (посредники) для выполнения транспортных служб.



http://

**Прокси-сервер** (проху — «представитель, уполномоченный») - промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника между пользователем и целевым сервером, позволяющий клиентам как выполнять косвенные запросы к другим сетевым службам, так и получать ответы.

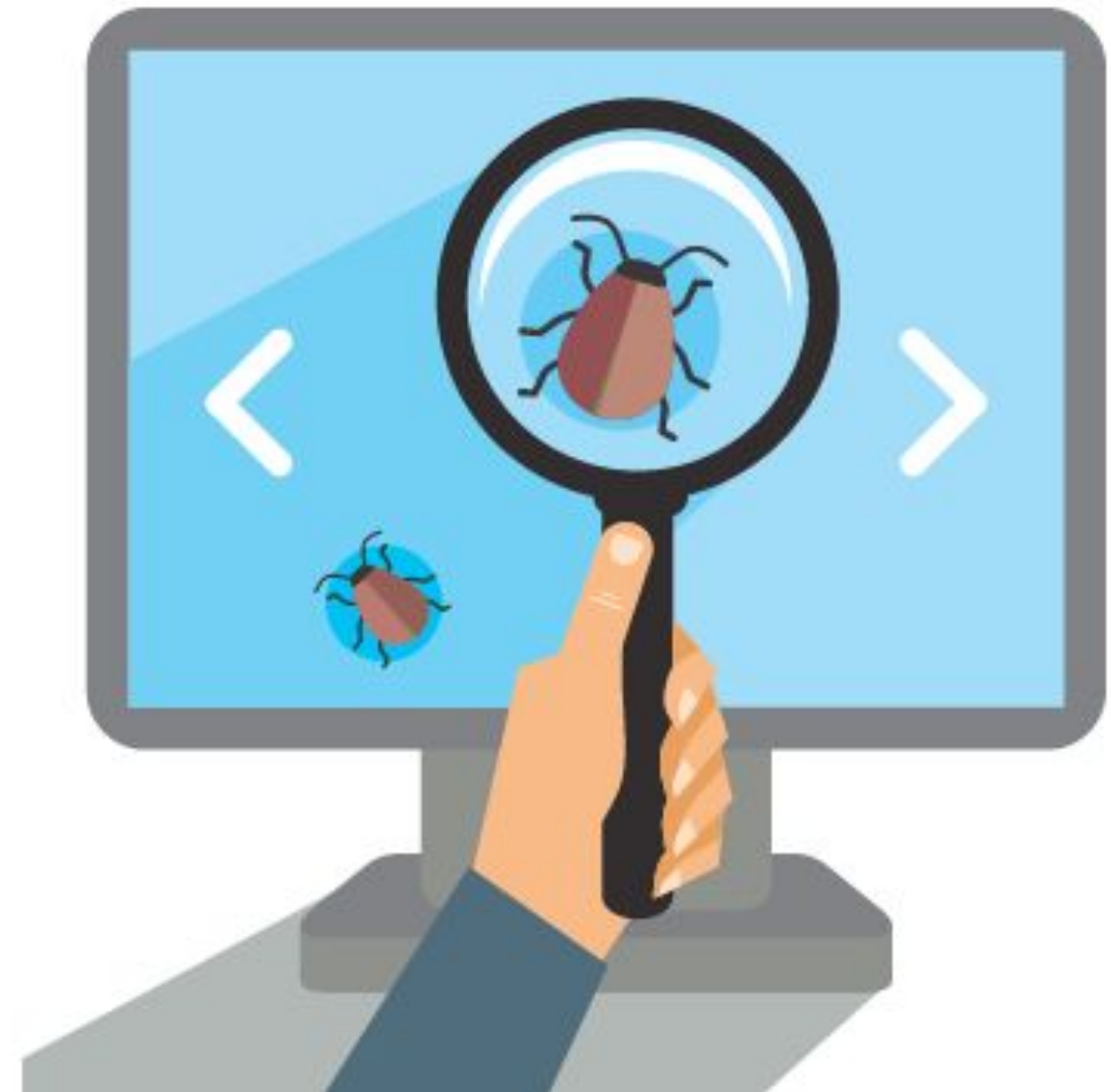






## Тестирование веб-приложений включает:

- Функциональное тестирование.
- Тестирование безопасности.
- Нагрузочное тестирование.
- Кроссбраузерное тестирование



**Функциональное тестирование** рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

**Преимущества функционального тестирования:**

- имитирует фактическое использование системы.

**Недостатки функционального тестирования:**

- возможность упущения логических ошибок в программном обеспечении;
- вероятность избыточного тестирования.

**Тестирование безопасности** - это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.



## Принципы безопасности программного обеспечения:

- **Конфиденциальность** - ограничение доступа к ресурсу некоторой категории пользователей или, другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.
- **Целостность и доверие** - ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.
- **Повреждение и восстановление** - в случае, когда данные повреждаются или неправильно меняются авторизованным или не авторизованным пользователем, Вы должны определить, насколько важной является процедура восстановления данных.
- **Доступность** - требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс, тем выше уровень доступности должен быть.

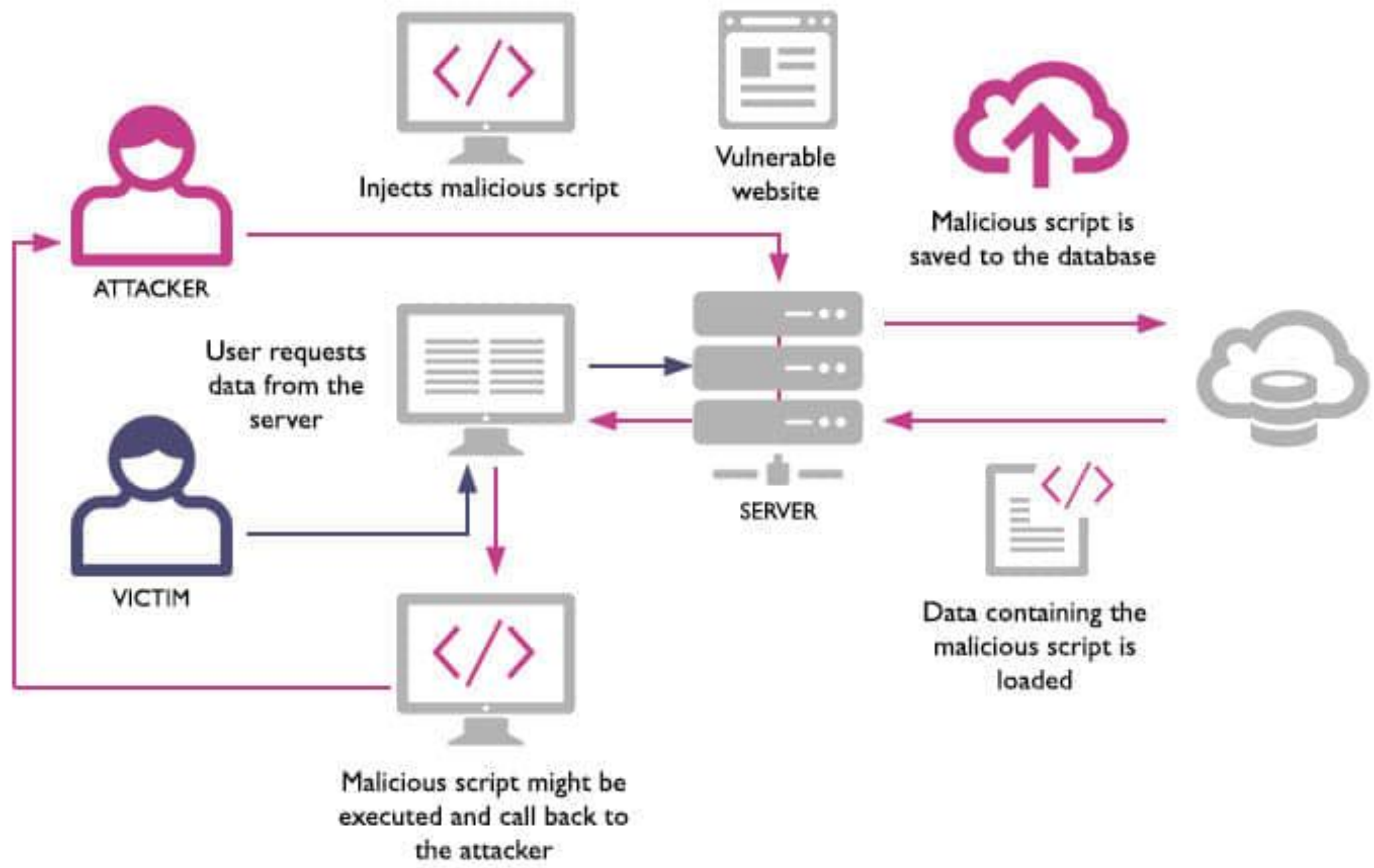
## Наиболее распространенные виды уязвимости:

- XSS (Cross-Site Scripting)
- XSRF / CSRF (Request Forgery)
- Code injections (SQL, PHP, ASP)
- Server-Side Includes (SSI) Injection
- Authorization Bypass



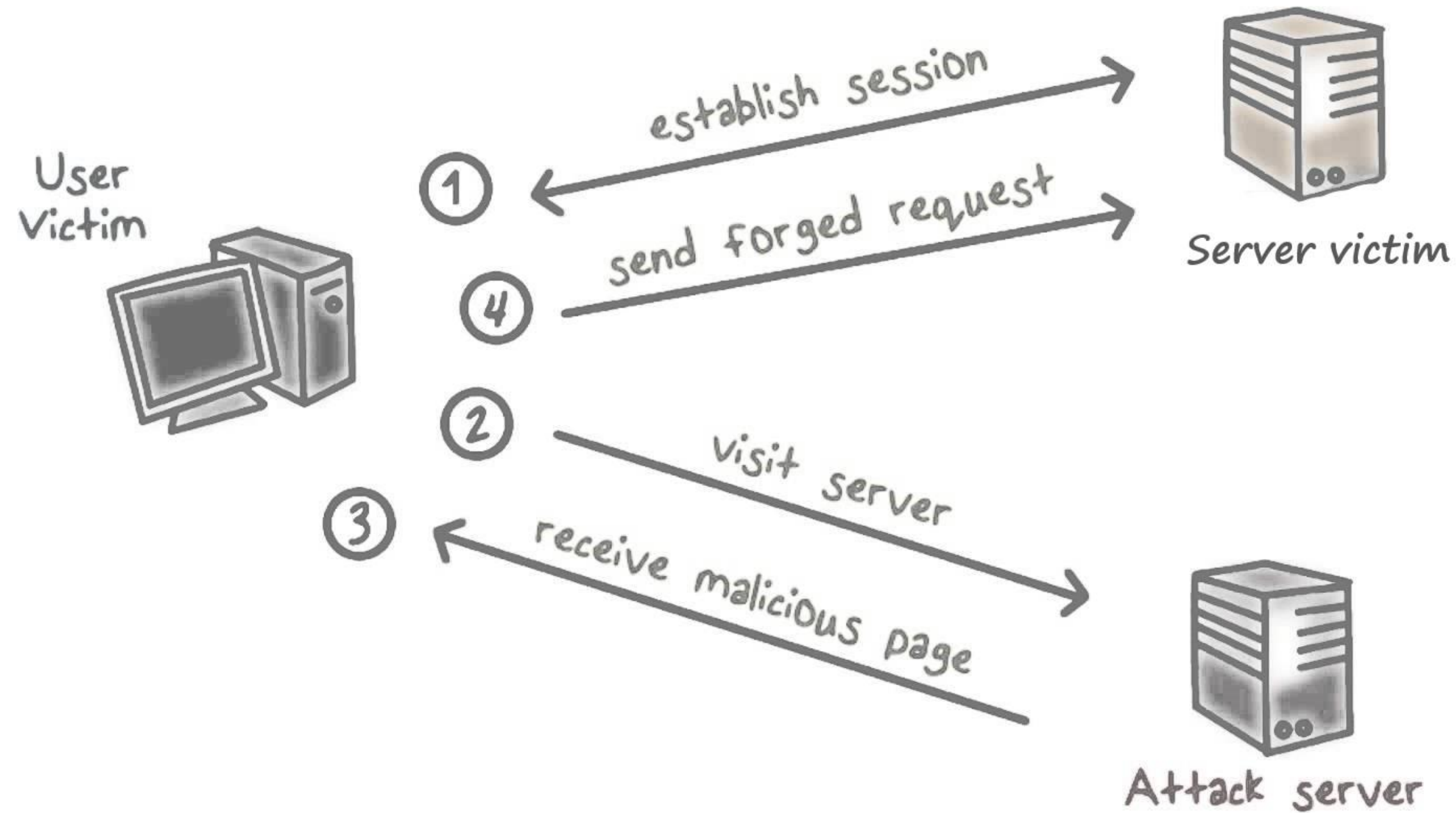


# XSS (Cross-Site Scripting)

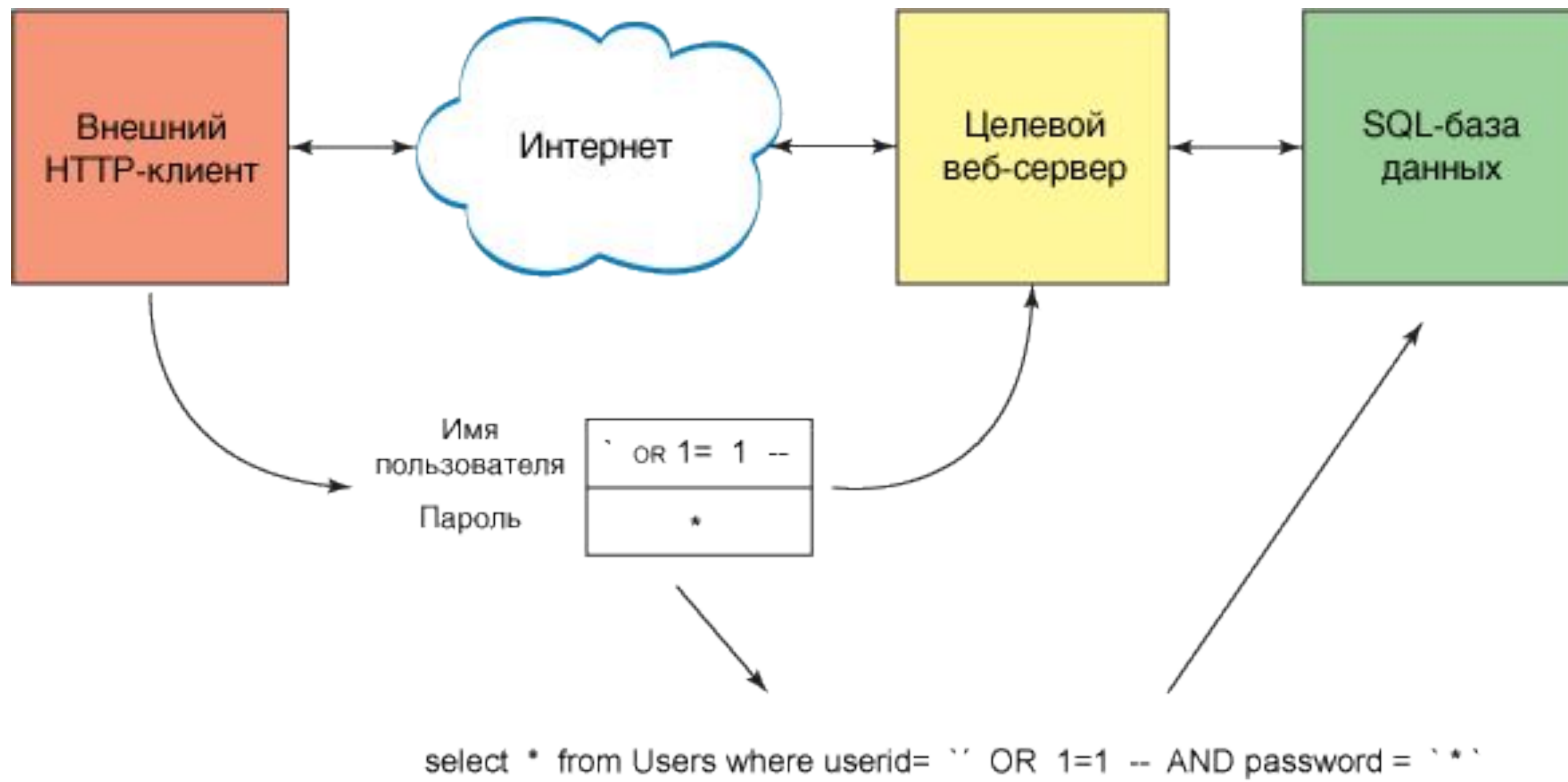


# XSRF / CSRF (Request Forgery)

## XSRF: Basic Idea

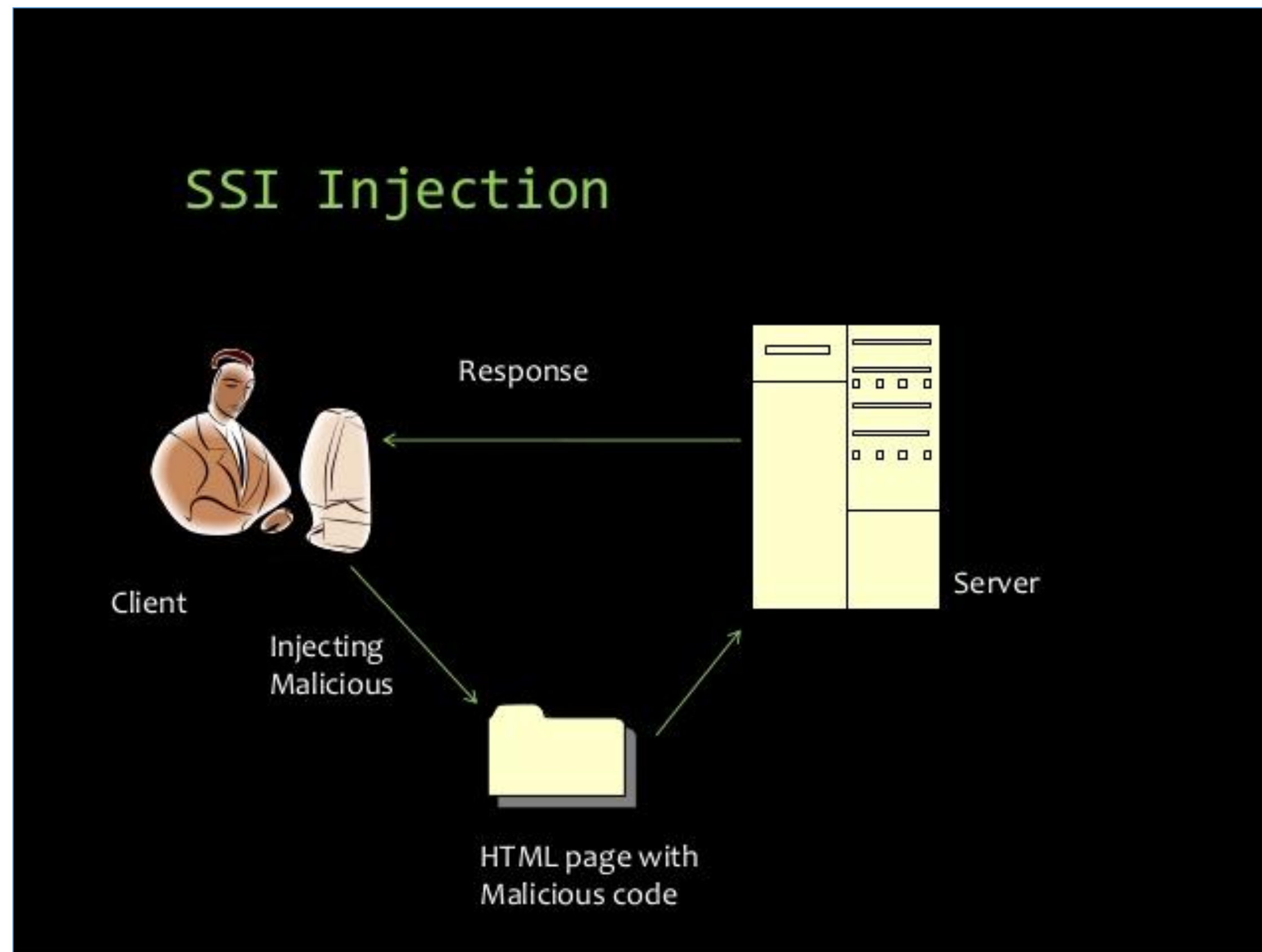


# SQL Injections

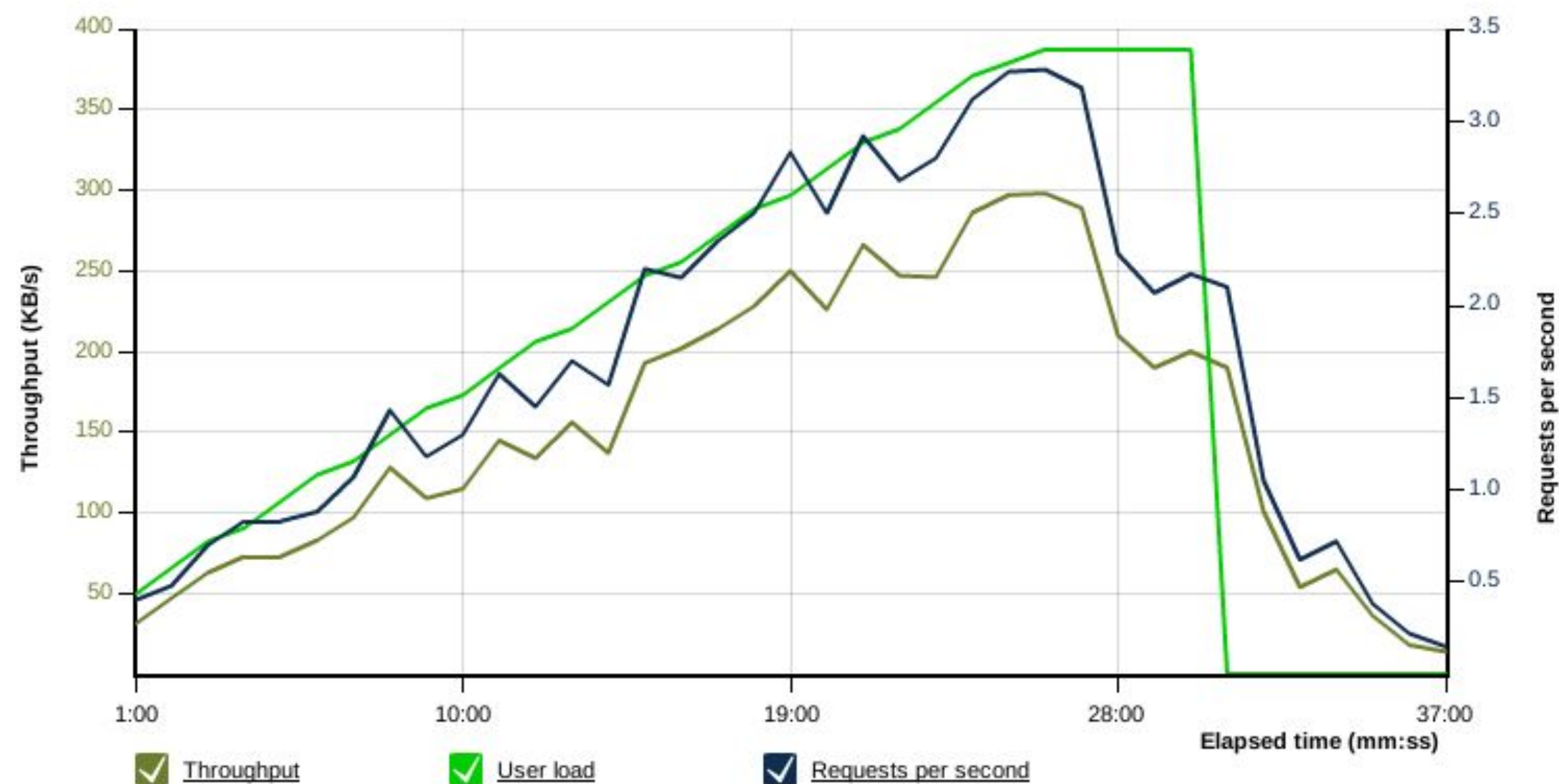




## Server-Side Includes (SSI) Injection



**Нагрузочное тестирование или тестирование производительности** - это автоматизированное тестирование, имитирующее работу определенного количества бизнес пользователей на каком-либо общем (разделяемом ими) ресурсе.



## Основные виды нагрузочного тестирования

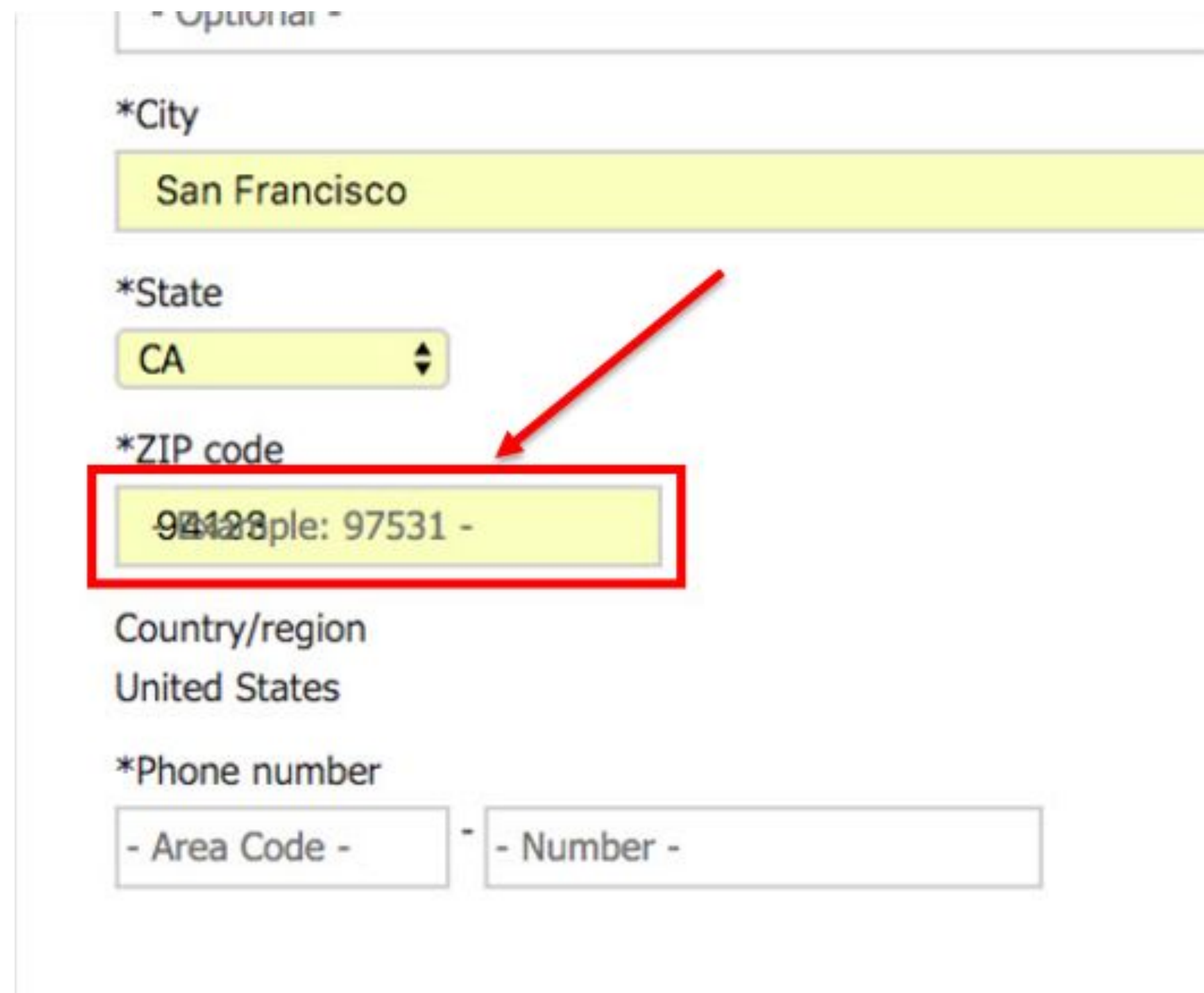
- Тестирование производительности (Performance testing)
- Стрессовое тестирование (Stress Testing)
- Объемное тестирование (Volume Testing)
- Тестирование стабильности или надежности (Stability / Reliability Testing)

**Кроссбраузерное тестирование** - вид тестирования, направленный на поддержку и правильное полное отображение программного продукта в разных браузерах, мобильных устройствах, планшетах, экранах различного размера.



## Часто встречающиеся дефекты:

- Верстка
- Навигация
- Ошибки JavaScript



The image shows a portion of a web form with several input fields. The fields are: \*City (text input with 'San Francisco'), \*State (dropdown menu with 'CA'), \*ZIP code (text input with '94128' and a red box around it), Country/region (text input with 'United States'), and \*Phone number (two text inputs: '- Area Code -' and '- Number -'). A red arrow points from the top right towards the ZIP code field.



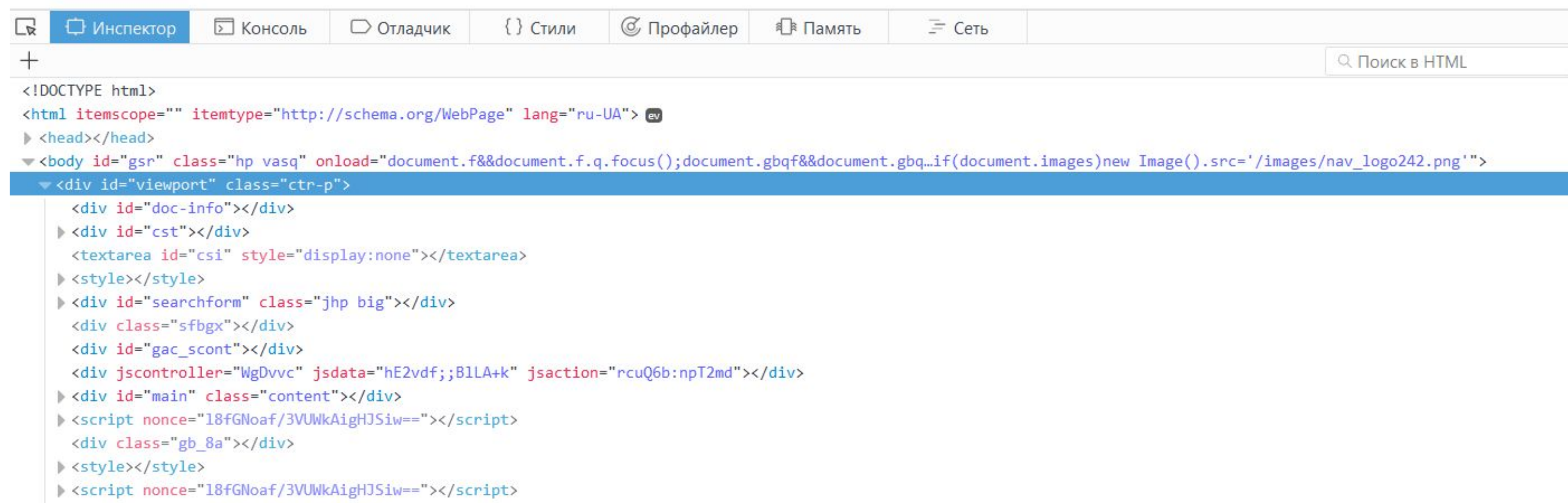
## Инструменты для кросс-браузерного тестирования:

- Browsershots
- Browser Sandbox
- Netrenderer
- Microsoft Edge
- Browsera
- Cross Browser Testing
- Browser Stack

The screenshot shows a web application interface for testing on various devices. On the left is a 'Quick Launch' sidebar with categories: Android, iOS, Windows Phone, Windows (versions 10, 8.1, 8, 7, XP), and Mac. The main area is a table of devices, grouped into columns for Samsung, Google, and Others. Each device entry includes a mobile icon, the device name, and a version number. Some entries have browser icons (Chrome, Firefox, Opera) next to them. At the bottom, there is a blue banner with a mobile icon, the text 'Test on physical devices! Look for this icon.', and a 'Got it' button.

Quick Launch	Samsung	Google	Others
<b>Android</b>	Galaxy S8 (7)	Pixel 2 (8)	<b>LG</b>
	Galaxy S8+ (7)	Pixel (8)	G5 (6)
<b>iOS</b>	Galaxy S7 (6)	Pixel XL (7.1)	<b>Motorola</b>
<b>Windows Phone</b>	Galaxy S6 (5)	Pixel (7.1)	Moto X 2nd Gen (6)
<b>Windows</b>	Galaxy S5 (4.4)	Nexus 6P (7)	Moto X 2nd Gen (5)
10	Galaxy S4 (4.4)	Nexus 5X (7)	Moto G 2nd Gen (5)
8.1	Galaxy Note 8 (7.1)	Nexus 6 (6)	<b>HTC</b>
8	Galaxy Note 4 (4.4)	Nexus 5 (4.4)	One M8 (4.4)
7	Galaxy Note 3 (4.3)	Nexus 9 (5.1 Tablet)	<b>Amazon</b>
XP	Galaxy Tab 4 10.1 (4.1)	Nexus 7 (6 Tablet)	Kindle Fire HDX 7 (4.3 Tablet)
<b>Mac</b>	Galaxy S3 (4.1)	Nexus 7 (4.4 Tablet)	Kindle Fire HD 8.9* (4 Tablet)
	Galaxy S2 (2.3)	Nexus 7 (4.2)	<b>Sony</b>
	Galaxy S (2.2)		Xperia Z5 (5.1)

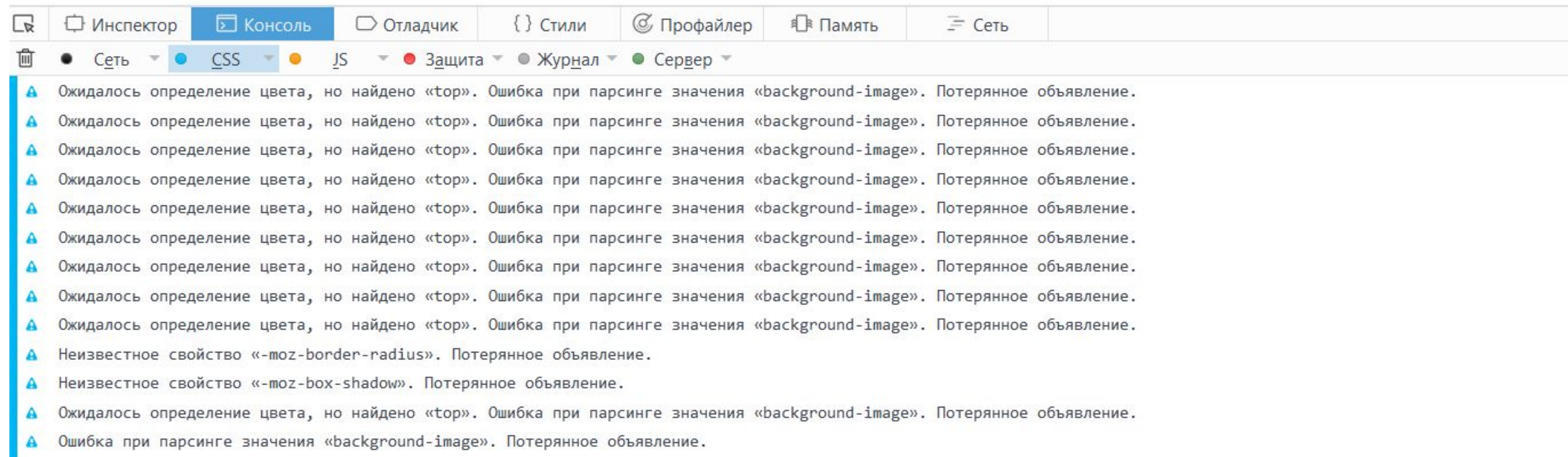
**Инспектор** - позволяет видеть HTML-код и CSS, который применён к каждому элементу на странице. Также позволяет в реальном времени редактировать как HTML, так и CSS. Внесённые изменения можно увидеть непосредственно в окне браузера.



```
Инспектор  Консоль  Отладчик  {} Стили  Профайлер  Память  Сеть
+
Поиск в HTML
<!DOCTYPE html>
<html itemscope="" itemtype="http://schema.org/WebPage" lang="ru-UA" >
  <head></head>
  <body id="gsr" class="hp vasq" onload="document.f&&document.f.q.focus();document.gbqf&&document.gbq...if(document.images)new Image().src='/images/nav_logo242.png'">
    <div id="viewport" class="ctr-p">
      <div id="doc-info"></div>
      <div id="cst"></div>
      <textarea id="csi" style="display:none"></textarea>
      <style></style>
      <div id="searchform" class="jhp big"></div>
      <div class="sfbgx"></div>
      <div id="gac_scont"></div>
      <div jscontroller="WgDvvc" jsdata="hE2vdf;;B1LA+k" jsaction="rcuQ6b:npT2md"></div>
      <div id="main" class="content"></div>
      <script nonce="18fGNoaf/3VUWkAighJSiw=="></script>
      <div class="gb_8a"></div>
      <style></style>
      <script nonce="18fGNoaf/3VUWkAighJSiw=="></script>
```



**Консоль** - инструмент, где выводятся сообщения и ошибки JavaScript, CSS и другие. Она позволяет загружать JavaScript вопреки порядку загрузки скрипта в браузере и докладывает об ошибках, как только браузер пытается выполнить Ваш код.





**Отладчик JavaScript** - инструмент для отладки JavaScript, если он не работает, как ожидалось. Он позволяет Вам загружать JavaScript вопреки порядку загрузки скрипта в браузере и докладывает об ошибках, как только браузер пытается ВЫПОЛНИТЬ КОД.



The screenshot shows the Chrome DevTools JavaScript Debugger interface. The top navigation bar includes icons for the Inspector, Console, Debugger (highlighted), Styles, Profiler, Memory, and Network. Below the navigation bar, the call stack on the left shows the current execution context: `cb=gapi.loaded_0` at `https://apis.google.com`. The main pane displays the source code of `gapi.loaded_0` with the following lines visible:

```
1 /* JS */ gapi.loaded_0(function(_){var window=this;
2 var ha,ia,ja,ma,sa,na,ta,ya,Ja;_.ea=function(a){return function(){return _.da[a].apply(this,arguments)}};_.DumpException=function(a){
3 ma=function(){var a=0;return function(b){return"jscomp_symbol_"+(b||"")+a++}}();sa=function(){ja();var a=ia.Symbol.iterator;a||(a=ia.S
4 _.wa=function(a){sa();var b=a[window.Symbol.iterator];return b?b.call(a):na(a)};_.xa="function"==typeof Object.create?Object.create:fu
5 Ja=function(a,b){if(b){var c=ia;a=a.split(".");for(var d=0;d<a.length-1;d++){var e=a[d];e in c||(c[e]={});c=c[e]}a=a[a.length-1];d=c[a
6 Ja("WeakMap",function(a){function b(a){Ka(a,d)||ha(a,d,{value:{}})}function c(a){var c=Object[a];c&&(Object[a]=function(a){b(a);return
7 1).toString();if(a){ja();sa();a=_.wa(a);for(var b;!b=a.next().done;)b=b.value,this.set(b[0],b[1])};f.prototype.set=function(a,c){b(
8 Ja("Map",function(a){if(function(){if(!a||"function"!=typeof a||a.prototype.entries||"function"!=typeof Object.seal)return!1;try{var
9 {};}this.Pe=f();this.size=0;if(a){a=_.wa(a);for(var b;!b=a.next().done;)b=b.value,this.set(b[0],b[1])};c.prototype.set=function(a,b)
10 a.ke.Pi,a.ke.head=null,this.size--,!0):!1};c.prototype.clear=function(){this.lf={};this.Pe=this.Pe.Pi=f();this.size=0};c.prototype.has
11 d;!d=c.next().done;)d=d.value,a.call(b,d[1],d[0],this)};c.prototype[window.Symbol.iterator]=c.prototype.entries;var d=function(a,c){
12 c.next,{done:!1,value:b(c)};c=null}return{done:!0,value:void 0}}),f=function(){var a={};return a.Pi=a.next=a.head=a,h=0;return c});
13 Ja("Set",function(a){if(function(){if(!a||"function"!=typeof a||a.prototype.entries||"function"!=typeof Object.seal)return!1;try{var
14 new window.Map;if(a){a=_.wa(a);for(var b;!b=a.next().done;)this.add(b.value)}this.size=this.V.size};b.prototype.add=function(a){this
15 b.prototype.values;b.prototype[window.Symbol.iterator]=b.prototype.values;b.prototype.forEach=function(a,b){var c=this;this.V.forEach(
16 _.Ma=function(a){var b=typeof a;if("object"==b)if(a){if(a instanceof Array)return"array";if(a instanceof Object)return b;var c=Object.
17 else if("function"==b&&"undefined"==typeof a.call)return"object";return b};_.Oa=function(a){return"array"==_.Ma(a)};_.Pa="closure_uid_
18 _.z=function(a,b){function c(){}c.prototype=b.prototype;a.H=b.prototype;a.prototype=new c;a.prototype.constructor=a;a.bp=function(a,c,
```



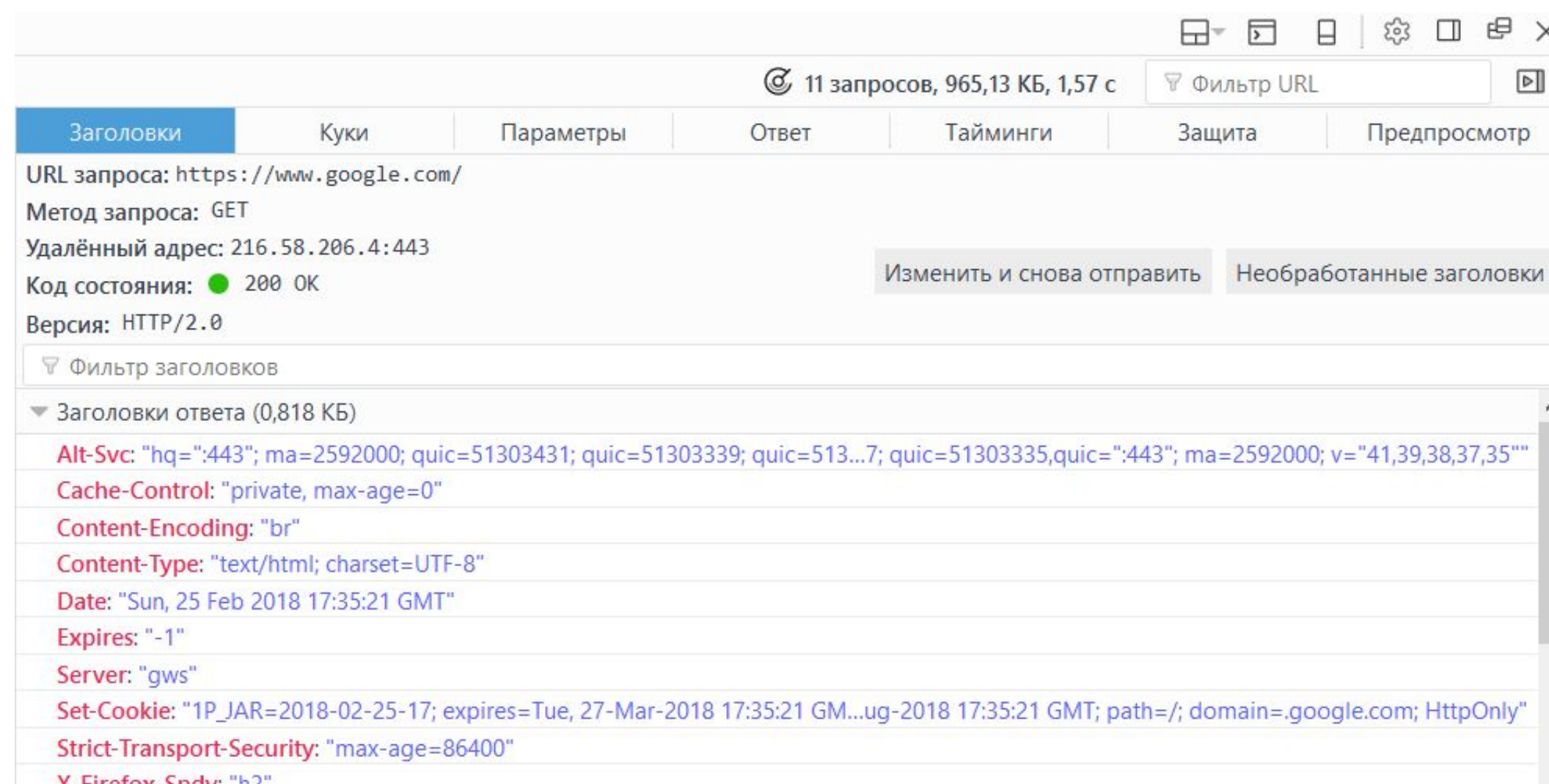
**Сеть** - записывает и отображает сетевые запросы в любое время, когда панель инструментов открыта, даже если сам монитор сети не выбран. Отображает запросы, методы, статус коды, объем данных.



Статус	Метод	Файл	Домен	Причина	Тип	Передано	Размер	0 мс
200	GET	/	www.google.com	document	html	58,23 КБ	193,39 КБ	
200	GET	rs=ACT90oFxLYDwuDJ4rYxoeOUZJB40O9szDg	www.google.com	script	js	141,03 КБ	407,82 КБ	
204	POST	gen_204?s=webaft&atyp=csi&ei=2fOSWtqDK4fWkwWDwJi...	www.google.com	beacon	html	—	0 6	
200	GET	rs=AA2YrTtMoJIMGQfOfyZyZ7reaiaiva99OQ	www.gstatic.com	script	js	46,40 КБ	135,49 КБ	
200	GET	rs=ACT90oEkGc7BMGCJQLR_QLFnTrvvrATVag?xjs=s1	www.google.com	script	js	28,58 КБ	88,78 КБ	
200	GET	rs=ACT90oEkGc7BMGCJQLR_QLFnTrvvrATVag	www.google.com	script	js	1,60 КБ	3,77 КБ	
200	GET	cb=gapi.loaded_0	apis.google.com	script	js	46,75 КБ	135,88 КБ	
200	GET	read	www.google.com.ua	xhr	html	—	0 6	
204	GET	write?data=&xsrif=ALAmJdEDTFXJk63_4iKDu0GbrBt5t2GQ9Q...	www.google.com	xhr	html	—	0 6	
204	POST	gen_204?atyp=csi&ei=2fOSWtqDK4fWkwWDwJioDQ&s=we...	www.google.com	beacon	html	—	0 6	

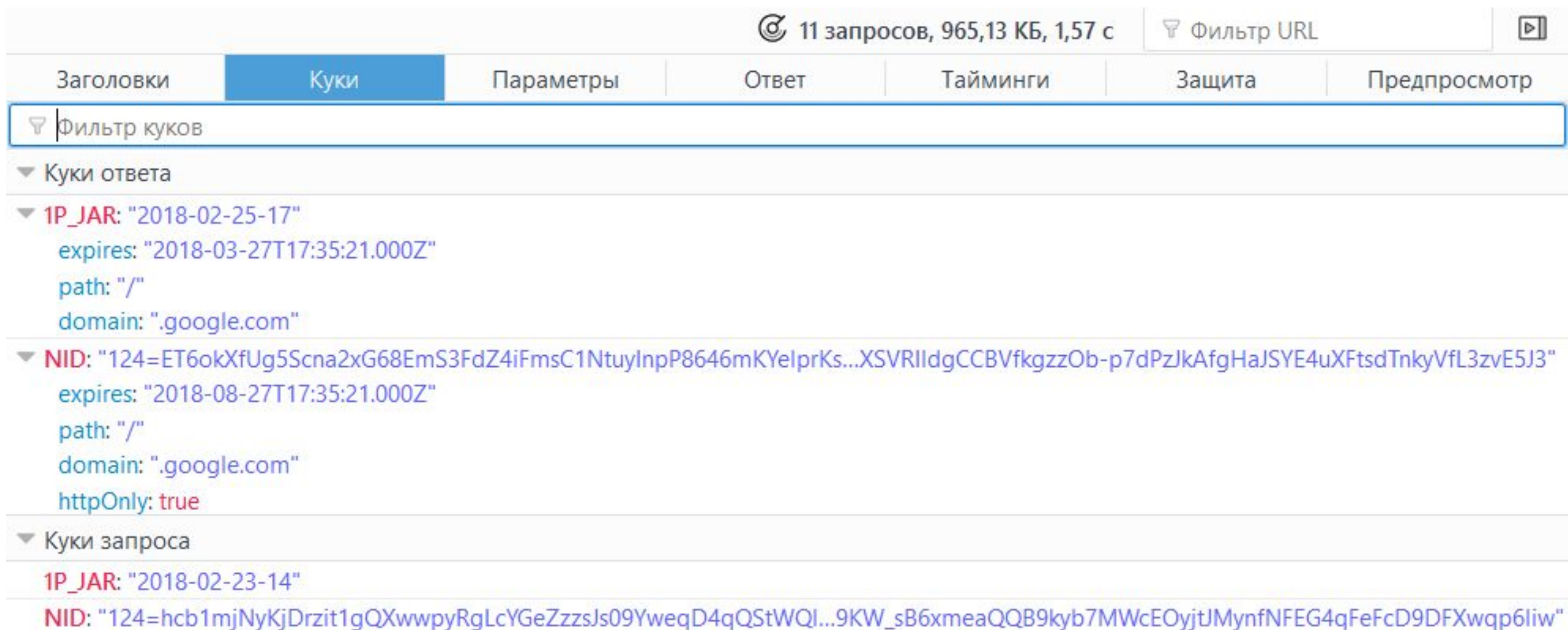
## Заголовки - перечислены основные сведения о запросе, в том числе:

- URL-адрес запроса.
- Метод запроса.
- Удаленный IP-адрес и порт.
- Код состояния.
- HTTP-запросы и заголовки ответов, которые были отправлены.





**Куки** - перечислены все сведения о любых файлах cookie, отправленных с запросом или ответом.



The screenshot shows the Chrome DevTools Network tab with the 'Cookies' sub-tab selected. The top bar indicates 11 requests, 965.13 KB, and 1.57 seconds. A search filter for 'Фильтр URL' is present. The 'Cookies' sub-tab is active, and a search filter for 'Фильтр куков' is visible. The cookies are organized into two sections: 'Куки ответа' (Response Cookies) and 'Куки запроса' (Request Cookies).

**Куки ответа**

- 1P\_JAR:** "2018-02-25-17"  
expires: "2018-03-27T17:35:21.000Z"  
path: "/"  
domain: ".google.com"
- NID:** "124=ET6okXfUg5Scna2xG68EmS3FdZ4iFmsC1NtuyInpP8646mKYelprKs...XSVRIldgCCBVfkgzzOb-p7dPzJkAfgHaJSYE4uXFtsdTnkyVfL3zvE5J3"  
expires: "2018-08-27T17:35:21.000Z"  
path: "/"  
domain: ".google.com"  
httpOnly: true

**Куки запроса**

- 1P\_JAR:** "2018-02-23-14"
- NID:** "124=hcb1mjNyKjDrzit1gQXwwpyRgLcYGeZzsj09YweqD4qQStWQI...9KW\_sB6xmeaQQB9kyb7MWcEOyjtJMynfNFEG4qFeFcD9DFXwqp6liw"

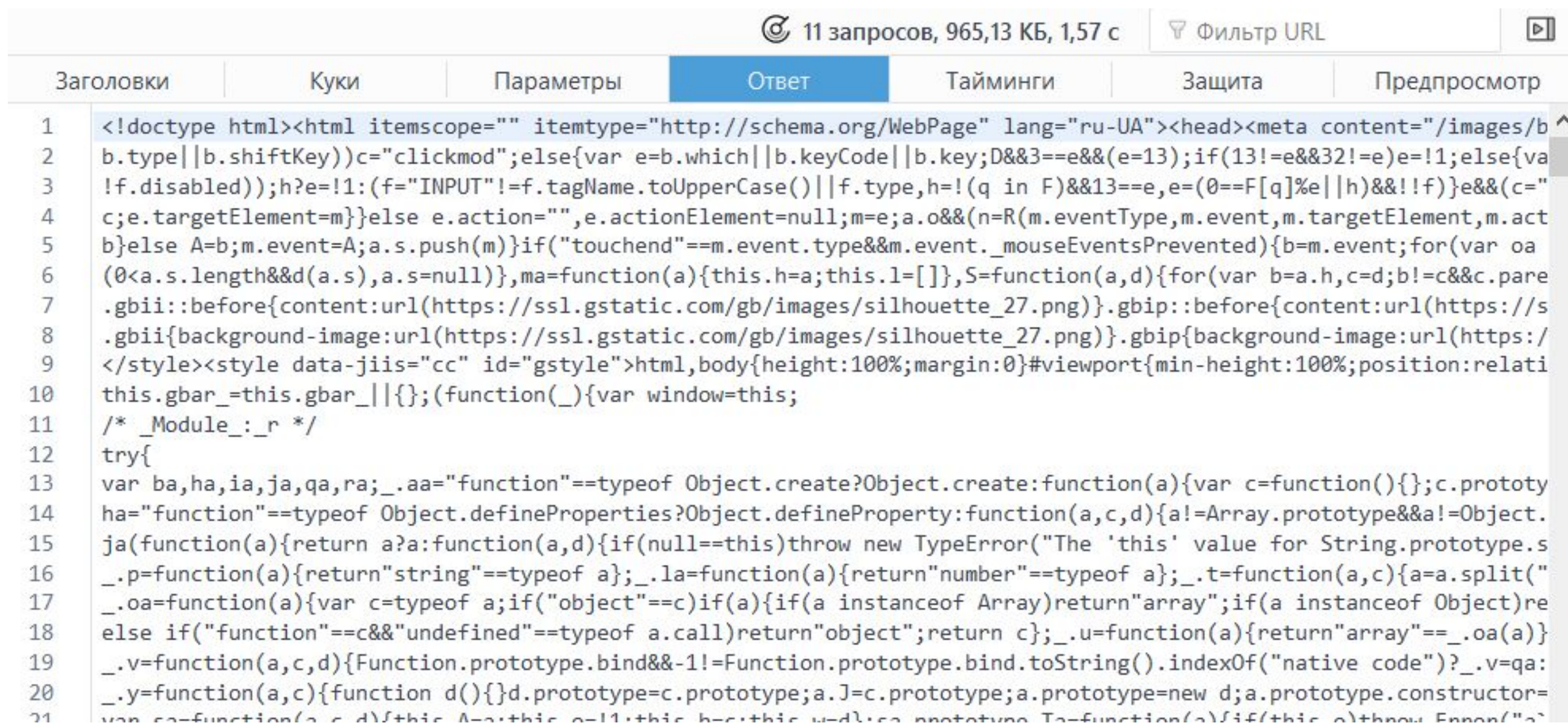
# Параметры - перечислены все параметры отправленные с запросом.

The screenshot shows the 'Parameters' tab in a web browser's developer tools. At the top, it displays '11 запросов, 965,13 КБ, 1,57 с' and a 'Фильтр URL' search box. Below this is a navigation bar with tabs: 'Заголовки', 'Куки', 'Параметры' (selected), 'Ответ', 'Тайминги', 'Защита', and 'Предпросмотр'. Under the 'Параметры' tab, there is a 'Фильтр параметров запроса' search box and a 'Строка запроса' section. The parameters listed are:

- s: "webaft"
- atyp: "csi"
- ei: "2fOSWtqDK4fWkwWDwJioDQ"
- rt: "wsrt.268,aft.159,prt.84"



# Ответ - отображается ответ пришедший на запрос в определенном формате данных.

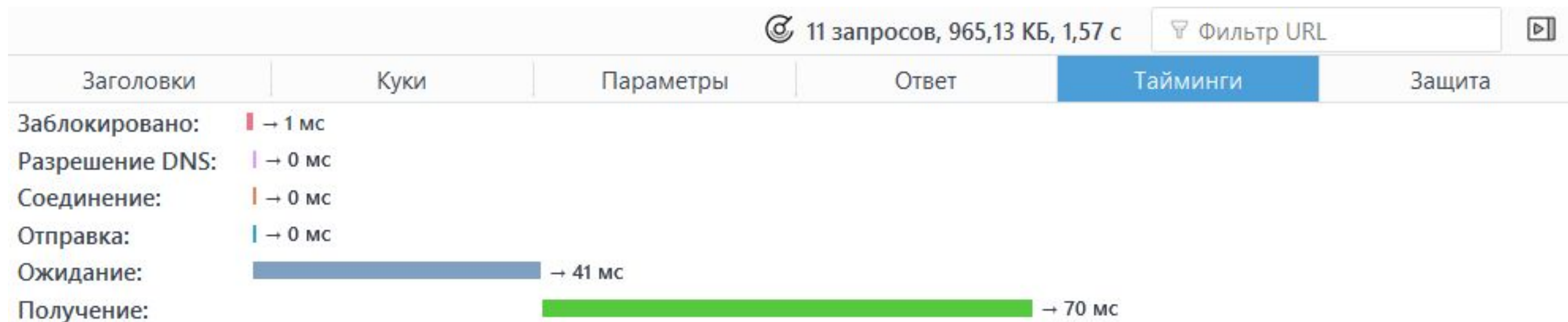


The screenshot shows the Chrome DevTools Network tab with the 'Response' pane selected. The response is an HTML document. The top part of the HTML includes a meta tag for content type and a script block. The script block contains a large amount of JavaScript code, including a try-catch block and several function definitions for creating and manipulating objects and arrays. The code is partially visible, showing lines 1 through 21.

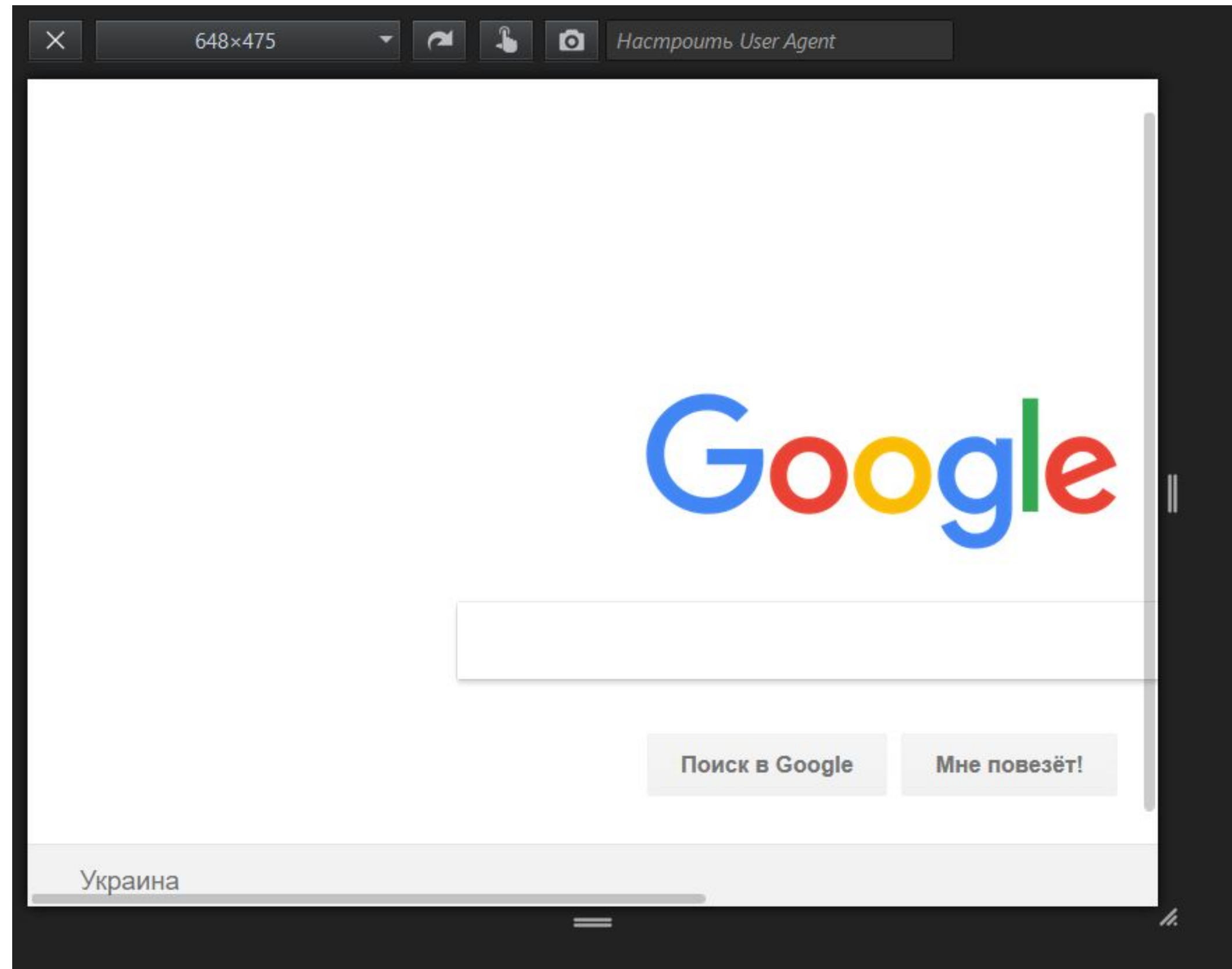
```
1 <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ru-UA"><head><meta content="/images/b
2 b.type||b.shiftKey))c="clickmod";else{var e=b.which||b.keyCode||b.key;D&&3==e&&(e=13);if(13!=e&&32!=e)e=!1;else{va
3 !f.disabled));h?e=!1:(f="INPUT"!=f.tagName.toUpperCase())||f.type,h!=(q in F)&&13==e,e=(0==F[q]%e||h)&&!f)}e&&(c="
4 c;e.targetElement=m}}else e.action="",e.actionElement=null;m=e;a.o&&(n=R(m.eventType,m.event,m.targetElement,m.act
5 b)else A=b;m.event=A;a.s.push(m)}if("touchend"==m.event.type&&m.event._mouseEventsPrevented){b=m.event;for(var oa
6 (0<a.s.length&&d(a.s),a.s=null)},ma=function(a){this.h=a;this.l=[]},S=function(a,d){for(var b=a.h,c=d;b!=c&&c.pare
7 .gbii::before{content:url(https://ssl.gstatic.com/gb/images/silhouette_27.png)}.gbip::before{content:url(https://s
8 .gbii{background-image:url(https://ssl.gstatic.com/gb/images/silhouette_27.png)}.gbip{background-image:url(https:/
9 </style><style data-jiis="cc" id="gstyle">html,body{height:100%;margin:0}#viewport{min-height:100%;position:relati
10 this.gbar_=this.gbar_||{};(function(_){var window=this;
11 /* _Module_:_r */
12 try{
13 var ba,ha,ia,ja,qa,ra;_aa="function"==typeof Object.create?Object.create:function(a){var c=function(){};c.prototy
14 ha="function"==typeof Object.defineProperty?Object.defineProperty:function(a,c,d){a!=Array.prototype&&a!=Object.
15 ja(function(a){return a?a:function(a,d){if(null==this)throw new TypeError("The 'this' value for String.prototype.s
16 _p=function(a){return"string"==typeof a};_la=function(a){return"number"==typeof a};_t=function(a,c){a=a.split("
17 _oa=function(a){var c=typeof a;if("object"==c)if(a){if(a instanceof Array)return"array";if(a instanceof Object)re
18 else if("function"==c&&"undefined"==typeof a.call)return"object";return c};_u=function(a){return"array"==_oa(a)}
19 _v=function(a,c,d){Function.prototype.bind&&-1!=Function.prototype.bind.toString().indexOf("native code")?_v=qa:
20 _y=function(a,c){function d(){d.prototype=c.prototype;a.J=c.prototype;a.prototype=new d;a.prototype.constructor=
21 var ca=function(a,c,d){this.A=this.a=1;this.b=this.ud=this.a.prototype.Ta=function(a){if(this.a)throw Error("a"
```



**Тайминги** - представлен более подробный аннотированный вид временной шкалы для каждого запроса, показывающий время выполнения.



**Режим адаптивного дизайна** - позволяет проверить сайт при разных разрешениях и сделать скриншот.





## Основные виды нагрузочного тестирования

- Тестирование производительности (Performance testing)
- Стрессовое тестирование (Stress Testing)
- Объемное тестирование (Volume Testing)
- Тестирование стабильности или надежности (Stability / Reliability Testing)

- Опишите двух и трехзвенные виды клиент-серверных архитектуры.
- Чем отличается веб-сайт от веб-сервиса?
- Что такое HTTP? Чем отличается от HTTPS?
- Что такое функциональное тестирование в контексте тестирования веб-приложений?
- Перечислите принципы тестирования безопасности и опишите и их.
- В чем заключается нагрузочное тестирование веб-приложений?
- Что проверяется при кросс-браузерном тестировании?
- Что такое инспектор кода и для чего он используется?
- Что такое кэш и куки и что в них хранится?
- Расскажите про знакомые вам инструменты тестирования веб-приложений.

СПАСИБ  
О!



Номер  
телефона



skype



E-mail