Cookie

**Ку́ки** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *cookie*, буквально — печенье) — небольшой фрагмент данных, отправленный [веб-сервером](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) и хранимый на [компьютере](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) пользователя. Веб-клиент (обычно [веб-браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80)) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе [HTTP](https://ru.wikipedia.org/wiki/HTTP)-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для[[1]](https://ru.wikipedia.org/wiki/Cookie#cite_note-microsoft-1):

* [аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) пользователя;
* хранения персональных предпочтений и настроек пользователя;
* отслеживания состояния [сеанса](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D0%B0%D0%BD%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1" \o "Сеанс (информатика) (страница отсутствует))[[en]](https://en.wikipedia.org/wiki/Session_(computer_science)) доступа пользователя;
* ведения статистики о [пользователях](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C). (Wiki)

Використанням модуля сookies.

<https://github.com/pillarjs/cookies>

npm install cookies

|  |  |  |
| --- | --- | --- |
| Підключенням модуля | var Cookies = require( "cookies" ) | var http = require('http')  var Cookies = require('cookies') |
|  | **cookies = new Cookies( request, response, [ options ] )** | // Optionally define keys to sign cookie values  // to prevent client tampering  var keys = ['keyboard cat']  var server = http.createServer(function (req, res) {  // Create a cookies object  var cookies = new Cookies(req, res, { keys: keys }) |
| Зчитування значення | cookies.get( name, [ options ] ) | // Get a cookie  var lastVisit = cookies.get('LastVisit', { signed: true }) |
| Встановлення значення | cookies.set( name, [ value ], [ options ] ) | // Set the cookie to a value  cookies.set('LastVisit', new Date().toISOString(), { signed: true })  if (!lastVisit) {  res.setHeader('Content-Type', 'text/plain')  res.end('Welcome, first time visitor!')  } else {  res.setHeader('Content-Type', 'text/plain')  res.end('Welcome back! Nothing much changed since your last visit at ' + lastVisit + '.')  }  })  server.listen(3000, function () {  console.log('Visit us at http://127.0.0.1:3000/ !')  }) |
| Видалення | cookies.set( "tampered", "baz", {expires: Date.now()-1}); |  |

If the options object is provided, it will be used to generate the outbound cookie header as follows:

* **maxAge**: a number representing the milliseconds from Date.now() for expiry
* **expires**: a Date object indicating the cookie's expiration date (expires at the end of session by default).
* **path**: a string indicating the path of the cookie (/ by default).
* **domain**: a string indicating the domain of the cookie (no default).
* **secure**: a boolean indicating whether the cookie is only to be sent over HTTPS (false by default for HTTP, true by default for HTTPS). [Read more about this option below](https://github.com/pillarjs/cookies" \l "secure-cookies).
* **httpOnly**: a boolean indicating whether the cookie is only to be sent over HTTP(S), and not made available to client JavaScript (true by default).
* **sameSite**: a boolean or string indicating whether the cookie is a "same site" cookie (false by default). This can be set to 'strict', 'lax', or true (which maps to 'strict').
* **signed**: a boolean indicating whether the cookie is to be signed (false by default). If this is true, another cookie of the same name with the .sig suffix appended will also be sent, with a 27-byte url-safe base64 SHA1 value representing the hash of cookie-name=cookie-value against the first [Keygrip](https://www.npmjs.com/package/keygrip) key. This signature key is used to detect tampering the next time a cookie is received.
* **overwrite**: a boolean indicating whether to overwrite previously set cookies of the same name (false by default). If this is true, all cookies set during the same request with the same name (regardless of path or domain) are filtered out of the Set-Cookie header when setting this cookie.

### Secure cookies

To send a secure cookie, you set a cookie with the secure: true option.

HTTPS is necessary for secure cookies. When cookies.set is called with secure: true and a secure connection is not detected, the cookie will not be set and an error will be thrown.

This module will test each request to see if it's secure by checking:

* if the protocol property of the request is set to https, or
* if the connection.encrypted property of the request is set to true.

==============================================================================================================

**Використання middleware cookie-parser**

<https://github.com/expressjs/cookie-parser>

npm install cookie-parser

|  |  |  |
| --- | --- | --- |
| Підключення | var express = require('express')  var cookieParser = require('cookie-parser')  var app = express() | var express = require('express')  var cookieParser = require('cookie-parser')  var app = express() |
| Встановлення як middleware | app.use( cookieParser(secret, options) )   * secret a string or array used for signing cookies. This is optional and if not specified, will not parse signed cookies. If a string is provided, this is used as the secret. If an array is provided, an attempt will be made to unsign the cookie with each secret in order. * options an object that is passed to cookie.parse as the second option. See [cookie](https://www.npmjs.org/package/cookie) for more information.   + decode a function to decode the value of the cookie | app.use(cookieParser(“Secret key”)) |
|  |  |  |
| Зчитування непідписаних | app.get('/', function (req, res) {  // Cookies that have not been signed  console.log('Cookies: ', req.cookies) |  |
| Зчитування підписаних | // Cookies that have been signed  console.log('Signed Cookies: ', req.signedCookies)  })  app.listen(8080) |  |
| Встановлення | **res.cookie**('name', 'value', {domain: '. . .', path: '. . .', . . . }); | **res.cookie**('name', 'Ivan', {domain: '.ex.com', path: '/start'});  res.cookie('name', 'express'); |
| Встановлення з вказаним періодом часу |  | //This cookie also expires after 360000 ms from the time it is set.  res.cookie(name, 'value', {maxAge: 360000}); |
| Видалення | **res.clearCookie('name');** | res.clearCookie('foo'); |

Приклад. Запам’ятовувати ім’я користувача.

|  |  |
| --- | --- |
|  |  |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**> </**head**> <**body**> <**form action="/login" method="post"**>  <**div**>  <**label**>Username:</**label**>  <**input type="text" name="username"**/>  </**div**>  <**div**>  <**label**>Password:</**label**>  <**input type="password" name="password"**/>  </**div**>  <**div**>  <**input type="submit" value="Log In"**/>  </**div**> </**form**>  </**body**> </**html**> | **var** express = require(**'express'**) **var** cookieParser = require(**'cookie-parser'**)  **var** app = express() app.use(express.**static**(\_\_dirname + **"/public"**));  app.use(cookieParser(**"My Top Secret String"**))  app.get(**'/'**, **function** (req, res) {  *// Cookies that have not been signed* **console**.log(**'Cookies: '**, req.**cookies**)   **if**(req.**cookies**.userName)  res.send(**"Hello "**+**req.cookies.userName**); //🡨 **Зчитування параметра**  **else** res.redirect(**"loginForm.html"**); }); *//----------------------------------------------------* **bodyParser** = require(**"body-parser"**); **const** urlencodedParser = **bodyParser**.urlencoded({**extended**: **false**}); app.post(**'/login'**,urlencodedParser ,**function** (req,res) {  **res.cookie('userName', req.body.username); //🡨 Встановлення значення**  res.redirect(**"/"**); });  app.get(**"/logout"**,**function** (req,res) {  **if**(req.**cookies**.userName)  {  **res.clearCookie("userName");** **//🡨 Видалення куки**  }  res.redirect(**"/"**) })  app.listen(3000,**function** () {  **console**.log(**"Listen 3000"**); }) |

==========================================================================================================================

**Сесії**

Веб-сервер не поддерживает постоянного соединения с клиентом, и каждый запрос обрабатывается, как новый, безо всякой связи с предыдущими. То есть, нельзя ни отследить запросы от одного и того же посетителя, ни сохранить для него переменные между просмотрами отдельных страниц. Вот для решения этих двух задач и были изобретены сессии.

Собственно, сессии, если в двух словах - это механизм, позволяющий однозначно идентифицировать браузер и создающий для этого браузера файл на сервере, в котором хранятся переменные сеанса.

Механизм сессий использует куки для хранения sessionid, и все. Куки хранятся на клиенте, сессия на сервере, следствия логичны - куки места на сервере не занимют, сессии недоступны для прямого чтения и изменения юзером. То есть в сессии вполне можно хранить пароль в открытом виде, например, хотя и не рекомендуется на всякий случай. Также размер куков ограничен, а в сессии вполне можно хранить довольно большие объемы данных, объекты и прочее.

Еще - сессия работает до закрытия браузера или до истечения серверного таймаута на хранение, куки можно поставить хоть на 10 лет. То есть, куки надо использовать для более долгих сроков (галка "запомнить меня", например).

[**https://github.com/expressjs/session**](https://github.com/expressjs/session)

npm install --save express-session

При використанні цього модуля сесії створюються автоматично для кожного користувача. Всі змінні при цьому додаються як властивості req.session.

|  |  |  |
| --- | --- | --- |
| Підключення | var session = require('express-session') | var express = require('express');  var cookieParser = require('cookie-parser');  var session = require('express-session'); |
|  |  | var app = express();  app.use(cookieParser());  app.use(session({ secret: 'its a secret', cookie: { maxAge: 60000 }, resave: false, saveUninitialized: false })); |
| Зчитування | req.**session. параметр** | app.get('/', function(req, res){  if(req.**session.page\_views**){ |
| Встановлення значення | req.**session. параметр = значення** | **req.session.page\_views**++;  res.send("You visited this page " + req.session.page\_views + " times");  } else {  **req.session.page\_views = 1**;  res.send("Welcome to this page for the first time!");  }  });  app.listen(3000); |
|  |  |  |
|  |  |  |
| Визначення, чи в перший раз користувач зайшов на сайт (якщо req.session.isNew дорівнює true) | req.session.isNew |  |
| Видалення сесії | **req.session = null** |  |

Приклад. Користувач може додавати в корзину товари (авто або телефон). Зберігати у корзині товари.

|  |  |  |
| --- | --- | --- |
| Шаблон index.ejs | ------------------ Форми ------------------------- | -------------------- app.js ---------------------------- |
| <**p**><%=title%></**p**> <**div**> <%if(**products**.**length**>0){%> <**ul**>  <%**products**.forEach((product)=>{%>  <**li**><%=**`**${product.**pr\_type**} **-** ${product.**title**}**`**%></**li**>  <%})%> </**ul**> <%}%> </**div**> <**div**>  <**a href="addAutoForm.html"**>Додати авто</**a**> </**div**> <**div**>  <**a href="addPhoneForm.html"**>Додати телефон</**a**> </**div**> | **------ Для додавання авто ---------**  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**> </**head**> <**body**> <**form action="/addAuto" method="post"**>  <**div**>  <**label**>Auto:</**label**>  <**input type="text" name="auto"**/>  </**div**>   <**div**>  <**input type="submit" value="Add auto"**/>  </**div**> </**form**>  </**body**> </**html**>  **------ Для додавання телефона -----**  <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**> </**head**> <**body**> <**form action="/addPhone" method="post"**>  <**div**>  <**label**>Phone :</**label**>  <**input type="text" name="phone"**/>  </**div**>   <**div**>  <**input type="submit" value="Add phone"**/>  </**div**> </**form**>  </**body**> </**html**> | **var** express = require(**'express'**); **const** path= require(**"path"**) **var** cookieParser = require(**'cookie-parser'**); **var** session = require(**'express-session'**);  **var** app = express();  app.use(cookieParser()); *//app.use(session({secret: "Shh, its a secret!"}));* app.use(session({ **secret**: **'its a secret'**, **cookie**: { **maxAge**: 60000 }, **resave**: **false**, **saveUninitialized**: **false** }));  app.set(**'views'**, path.join(\_\_dirname, **'views'**)); app.set(**'view engine'**, **'ejs'**) app.use(express.**static**(\_\_dirname + **"/public"**));  app.get(**"/"**,**function** (req,res) {  **if**(req.**session**.isNew || !req.**session**.**products**)  {  req.**session**.**products**=[];  res.render(**"index"**,{**title**:**"Корзина пуста"**,**products**:[]});   }  **else** {  res.render(**"index"**,{**title**:**"Корзина пуста"**,**"products"**:req.**session**.**products**});  } }); *//---------------------------------------------* **bodyParser** = require(**"body-parser"**); **const** urlencodedParser = **bodyParser**.urlencoded({**extended**: **false**}); app.post(**'/addAuto'**,urlencodedParser ,**function** (req,res) {  req.**session**.**products**.push({**pr\_type**:**"Auto"**, **title**:req.**body**.**auto**})  res.redirect(**"/"**); });  app.post(**'/addPhone'**,urlencodedParser ,**function** (req,res) {  req.**session**.**products**.push({**pr\_type**:**"Phone"**, **title**:req.**body**.**phone**})  res.redirect(**"/"**); }); app.listen(3000,**function** () {  **console**.log(**"Listen 3000"**); }) |
|  |  |  |