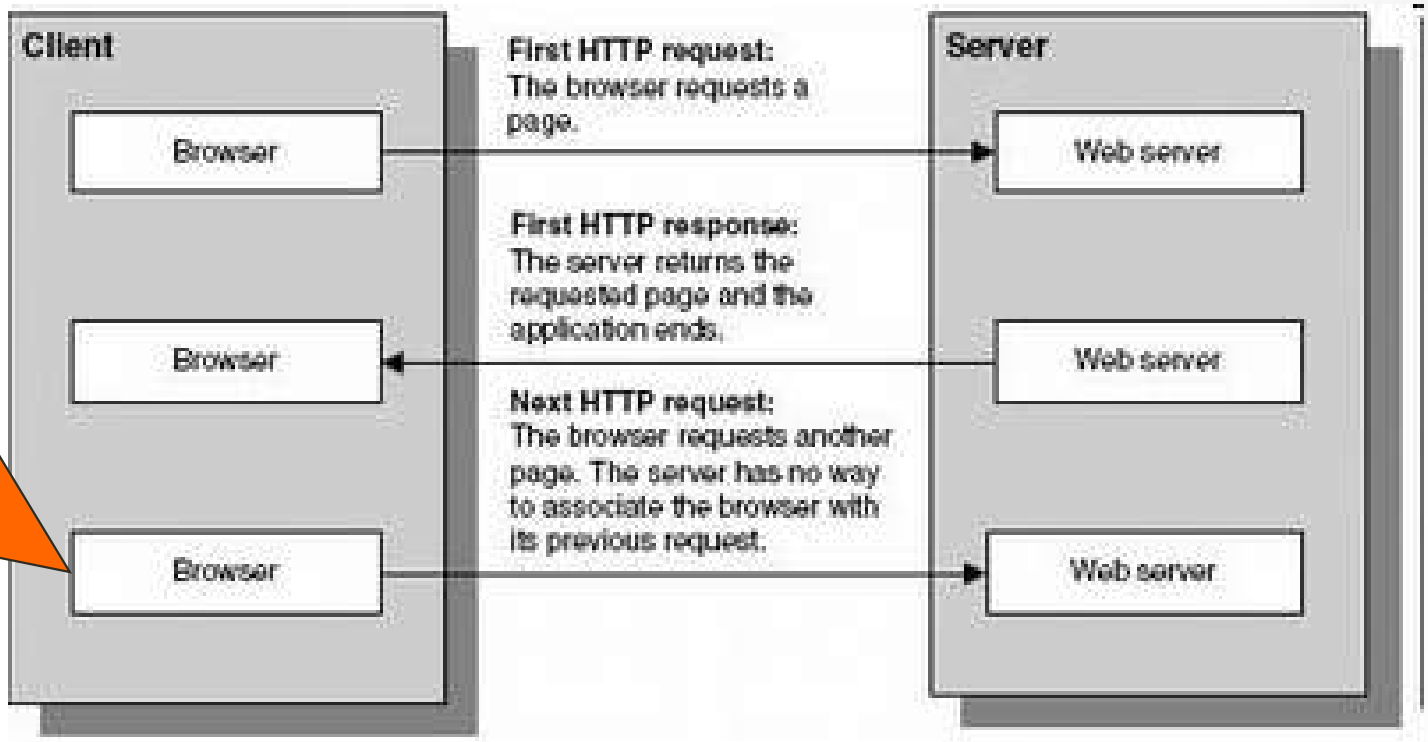
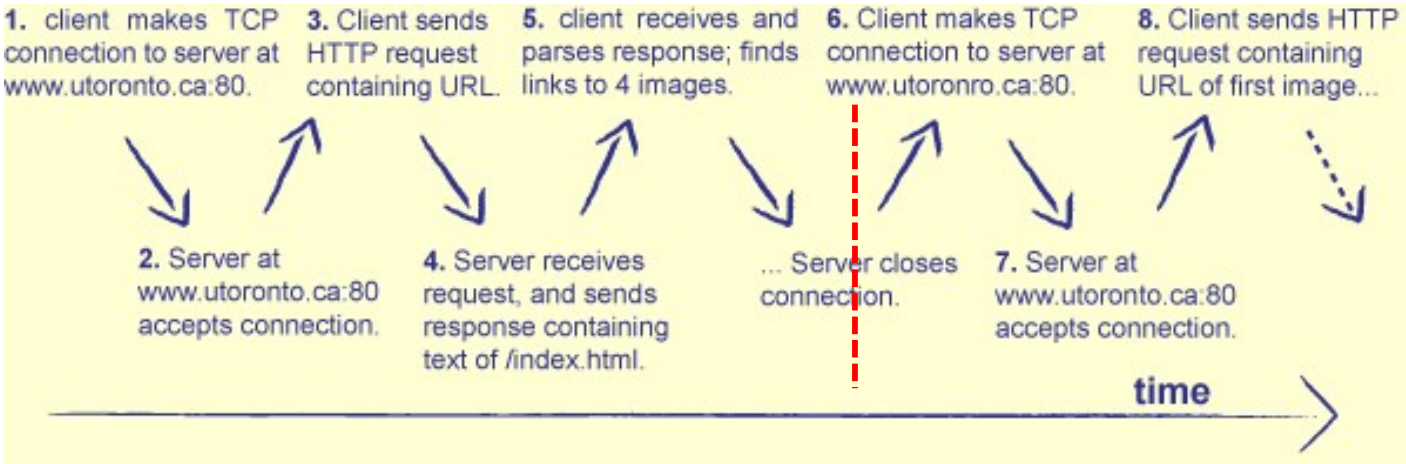
SESIONES Y COOKIES CON NODE.JS

Fernando Saez

HTTP es stateless

El servidor no tiene forma de asociar que este request es producido por el mismo navegador que el request anterior





Entendiendo Información de Estado del usuario

- Páginas web customizadas (personalizadas para un individuo) sobre sus preferencias.
- Información almacenada temporariamente para un usuario, Por Ej. Navegación sobre un form multiparte.
- Permitir a un usuario crear un bookmarks para retornar a ubicaciones específicas dentro del sitio.
- Proveer un carrito de compras para almacenar información sobre pedidos de compras.
- **Mantener estado significa almacenar información de manera temporal o persistente** acerca de los visitantes del sitio.

Opciones para almacenar información de estado

En el servidor	En el cliente
Estado de sesión • La información sólo está disponible para el usuario que inicio la sesión	**Cookies** • Archivos de texto que guardan información de estado
Base de datos • En algunos casos se puede usar una BD para guardar el estado	**Campos Hidden** • Retienen valores simples entre peticiones POST o GET
Archivos, xml	**Query strings** • Información agregada al final de la URL
	LocalStorage - SessionStorage

Datos en el front-end

- Campos hidden

```
<input type=“hidden” name=“miCampoOculto”>
```

- Querystring

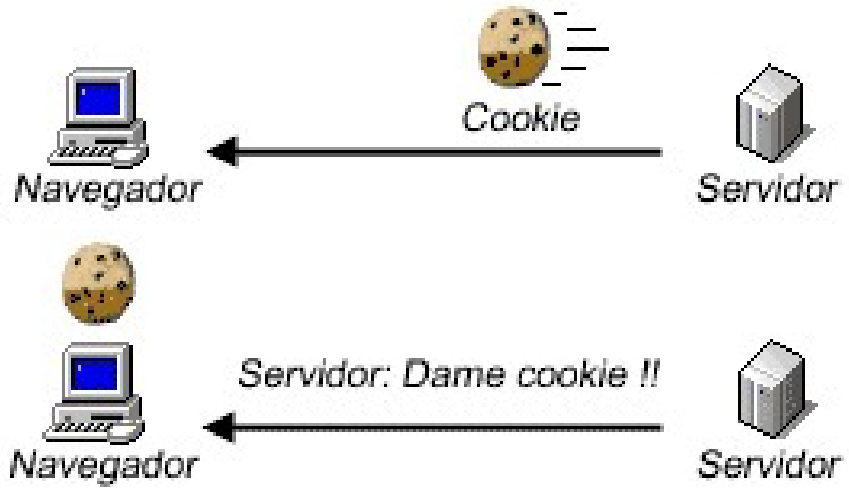
```
<A href=“def0.php?MiVar=unValor&MiOtraVar=otroValor">
Hacer click </a>
```

- LocalStorage – SessionStorage

```
localStorage.setItem('Nombre', 'Miguel Antonio')
localStorage.Apellido = 'Márquez Montoya'
```

Cookies

- Es un pequeño archivo de texto que el servidor envia al browser, para ser enviado luego entre los distintos request.
- El browser lo almacena en la máquina cliente de manera persistente.



https://youtu.be/I01XMRo2ESg

Cookies en Node.js

- $ npm install cookie-parser

```
var express = require('express')
var cookieParser = require('cookie-parser')
var app = express()
app.use(cookieParser())
app.get('/', function (req, res) {
  // Cookies no firmadas
  console.log('Cookies: ', req.cookies)
  // Cookies firmadas
  console.log('Signed Cookies: ', req.signedCookies)
})
app.listen(8080)
```

Ejemplo: Crear una cookie

```
router.post('/login', function(req, res, next) {
    res.cookie('mail', req.body.mail,{ expires: new
Date(Date.now() + (60*60*24*365*3)) });
    var pagina='<!doctype html><html><head></head><body>'+
            '<p>Se creo la cookie</p></body></html>';
    res.send(pagina);
});
router.get('/login', function(req, res, next) {
  if (req.cookies.mail)
    res.render('login',{mail:req.cookies.mail});
  else
    res.render('login');
});
```

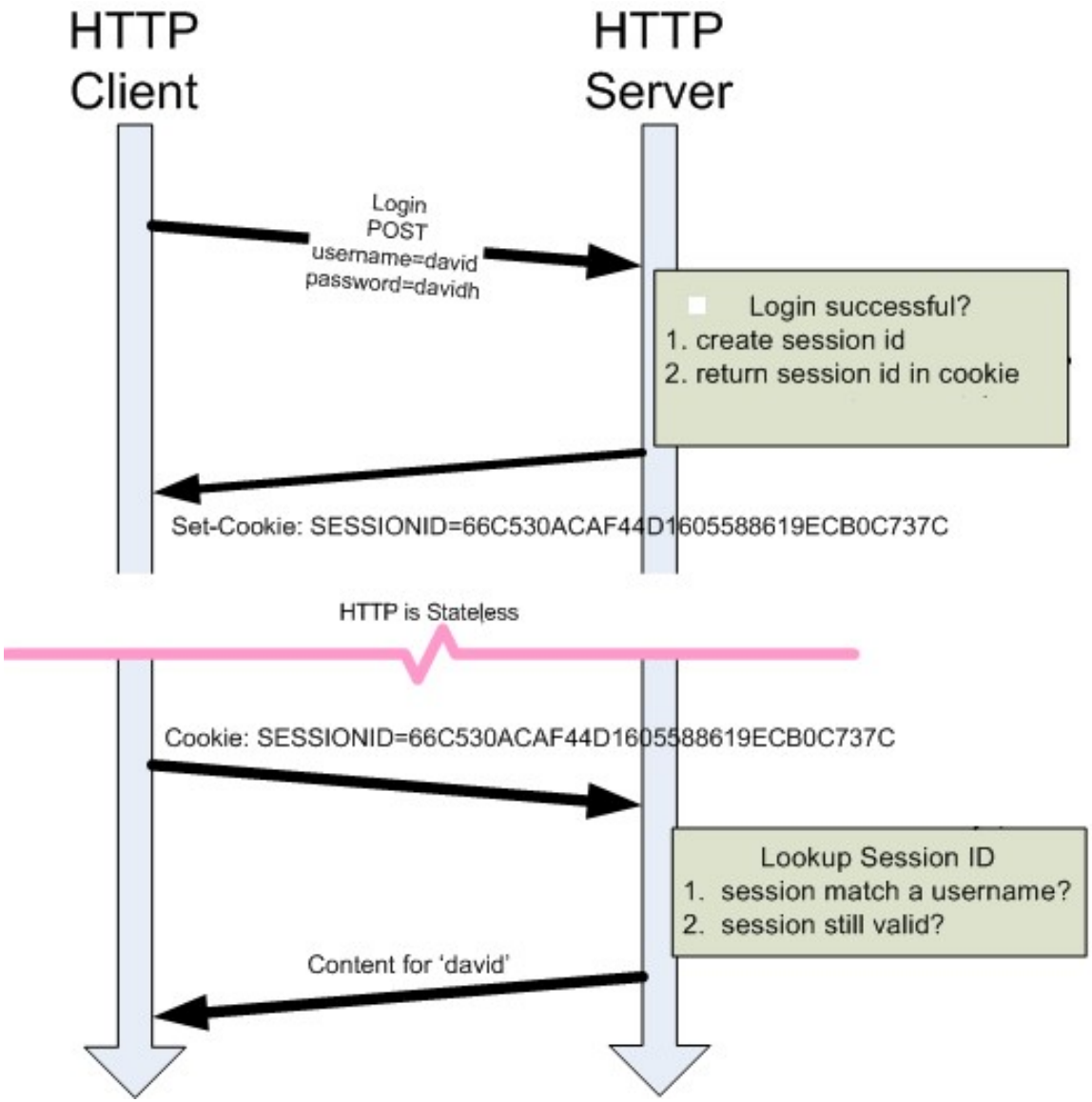
Concepto de Sesión

- Una sesión inicia cuando el visitante ingresa por primera vez a un sitio o aplicación Web, o después de que ese mismo visitante ha finalizado una sesión previa.

- El objetivo de definir este concepto de sesión es poder almacenar información que sobrevive las peticiones HTTP de un visitante.

- La sesión involucra una cookie con una fecha de expiración

Sesiones en Node



Datos de la sesión en cookies

- $ npm install cookie-session

```
var cookieSession = require('cookie-session')
var express = require('express')
var app = express()
app.use(cookieSession({
  name: 'session',
  secret: 'una cadena aleatoria',
// Opciones de la cookie
  maxAge: 24 * 60 * 60 * 1000 // 24 horas
}))
```

Acceder a Datos de la sesión en cookies

```
app.get('/', (req, res, next) => {
        req.session.views = (req.session.views || 0) + 1
// escribir respuesta
        res.send('Has visitado la página: ' +
                req.session.views + ' veces' );
});
```

- Otras propiedades útiles:

.isChanged - .isNew - .isPopulated

- Destruir una sesión

```
req.session = null
```

Datos de la sesión en el servidor

- $ npm install express-session

```
var session = require(‘express-session')
var app = express()
app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: true,
  //cookie: { secure: true } /*Solo usar si utilizamos cookies*/
}))
```

Datos de la sesión en el servidor

- Session.save(callback)

```
req.session.save(function(err) {
  // session guardada
})
```

- Session.reload(callback)

```
req.session.reload(function(err) {
  // session actualizada
})
```

- Session.destroy(callback)

```
req.session.destroy(function(err) {
  // No se puede acceder a la session aquí
})
```

Ejemplo de uso de express-session

```
const express = require('express');
const session = require('express-session');
const app = express();
app.use(session({
  secret: 'secret',
  resave: true,
  cookie: {
    maxAge: 24 * 60 * 60 * 365 * 1000
  }
}))
```

```
app.use(express.json());
app.use(express.urlencoded());
app.get('/', (req, res) => {
  if (req.session.views) {
    req.session.views++;
  }
  else {
    req.session.views = 1;
  }
  res.send(`${req.session.views}
views`);
})
app.listen(3000);
```