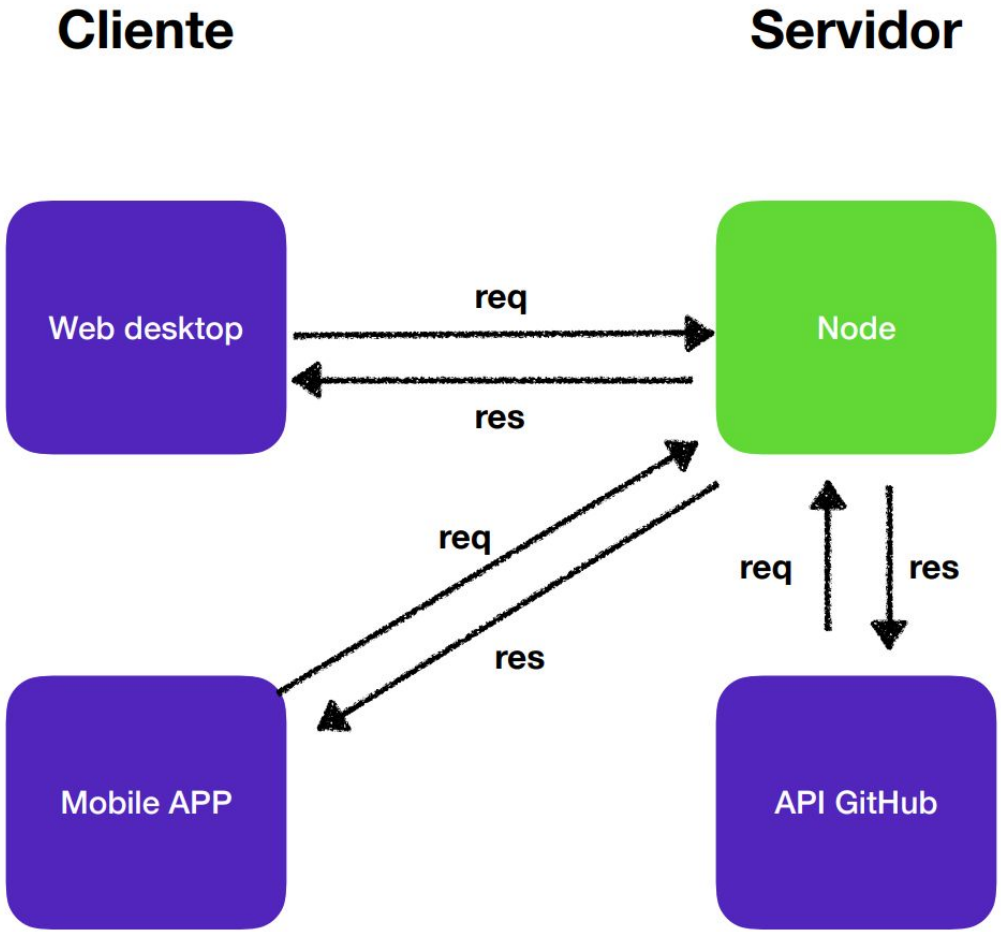
Desarrollo Backend con NodeJS

Clases, POO en Javascript y Agregando controllers en nuestra API

course[It]

Repasito:


Cliente
Servidor
Web desktop
req
res
Node
req
res
req
res
Mobile APP
API GitHub

Rutas

También los van a escuchar mencionados como **endpoints** por su nombre en inglés.

Son el **punto de entrada** de nuestra aplicación. Esto quiere decir que va a ser el lugar por donde vamos a recibir las llamadas HTTP de quien nos consuma

Pueden ser estáticas o dinámicas

Rutas dinámicas

Son rutas que aceptan una porción variable en su estructura. Esto implica que con un solo endpoint podemos manejar múltiples casos

- Por ejemplo: En los usuario de una red social o en los productos de un e-commerce

```
router.get("/users/:name", function (req, res, next) {
  res.send(`Hola usuario: ${req.params.name}`);
});

si la utilizamos así:
// http://localhost:3000/users/bel

Obtenemos:

// Hola usuario: Bel
```

req.query y req.params

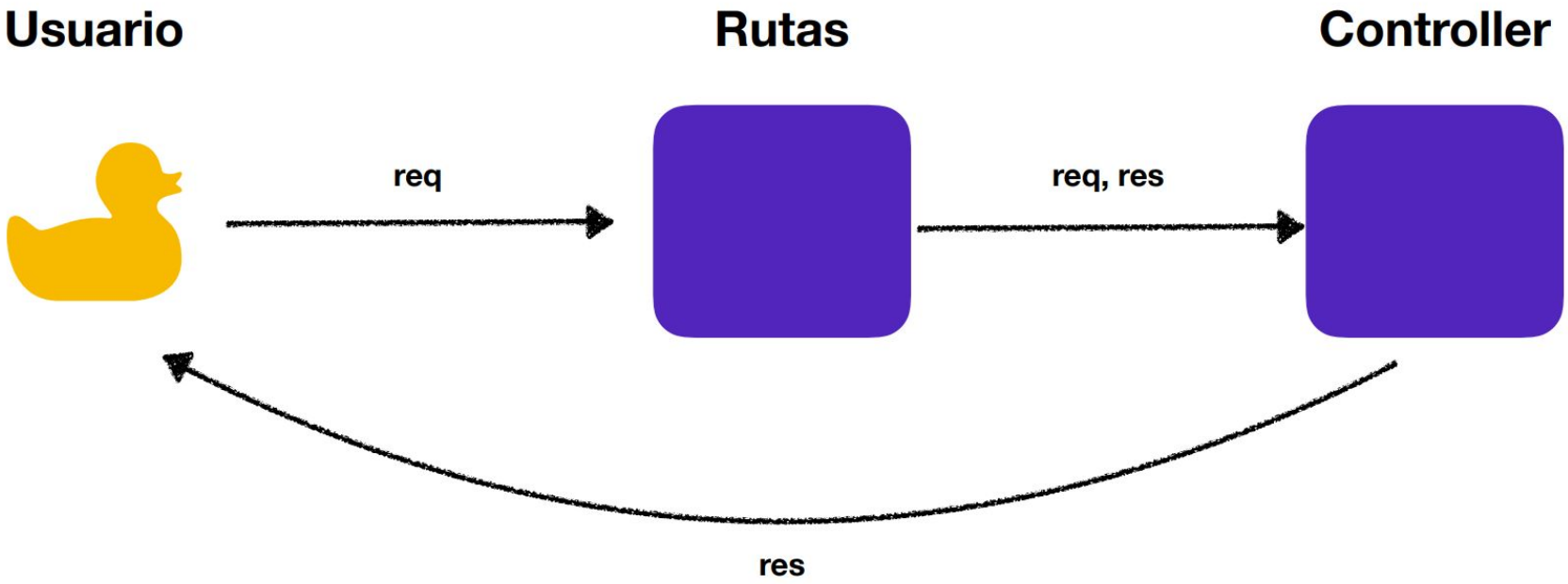
Son propiedades del objeto req que nos permiten acceder a información en la url

- req.params nos da acceso a la porción variable de una ruta dinámica (los controlamos desde el servidor)

- req.query nos da acceso a los parámetros que se envíen a través de la url a la ruta (vienen desde el cliente, no los controlamos desde el servidor)

req.query y req.params

```
router.get("/users/:name", function (req, res, next) {
  res.send(
    `Hola usuario: ${req.params.name} tiene ${req.query.gatos} gatos y ${req.query.perros} perros`
  );
});
```



Controllers

Los controllers nos proponen una capa extra para no tener todo nuestra lógica en el cuerpo de la ruta. Esto nos permite generar código más limpio, simple de leer, y más seguro.

En los controllers vamos a almacenar toda la lógica de negocio de nuestra aplicación

Programación orientada a Objetos... en Javascript

Javascript es un lenguaje multiparadigma que desde ES6 soporta clases. Una clase es un template para crear objetos que encapsulan variables y lógica para ser utilizada dentro del contexto de esa misma clase.

Una clase también puede exponer información de forma controlada

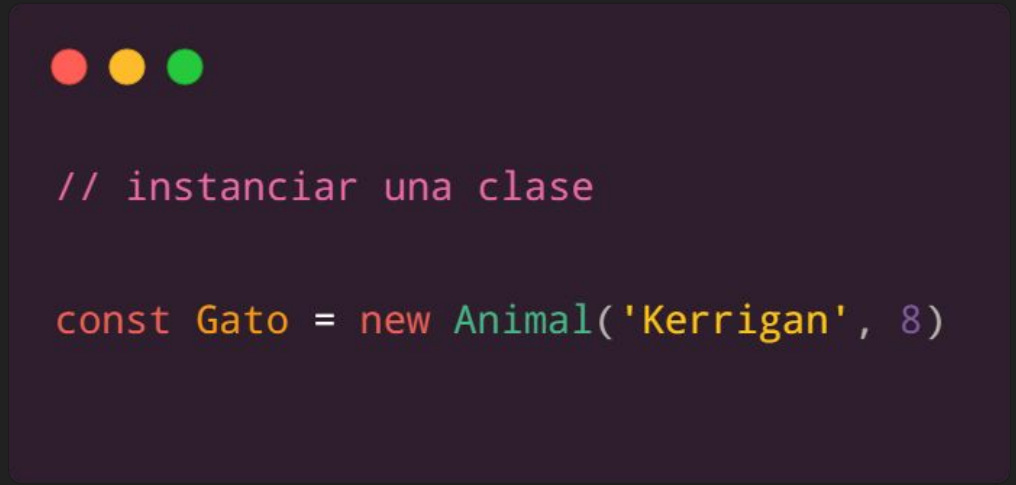
Sintaxis de una clase

```
// declarar una clase:

class Animal {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  ageUp() {
    return this.age++
  }
}
```

Sintaxis de una clase



Herencia

Cuando una clase hereda de otra recibe las propiedades de su clase madre, pero a su vez también puede agregar funcionalidad específica

```
class Cat extends Animal {
  constructor(name) {
    super(name); // necesitamos llamar a la palabra reservada super para indicarle
                 //que esta propiedad se hereda de la clase madre
  }

  speak() {
    console.log(`${this.name} meows.`);
  }
}

let Gato = new Cat('Alyx');
Gato.speak(); // Alyx meows.
```

Ahora sí, finalmente a hacer lo que vinimos a hacer

- `npx express-generator nombreDelDirectorioDeseado --no-view`
  - http://expressjs.com/en/starter/generator.html#express-application-generator