* Argumenten aan de next functie in de middleware:
  + Er wordt geen waarde aan een andere functie gegeven, maar er wordt een nieuwe error teruggegeven.
  + Je kunt er een error string of een object erin stoppen.
* Error handling binnen een specifieke functie
  + App.use(‘/birds’, function(err, req, res, next)) {
    - Const error = new Error(“error”);
    - Error.name = “gekkeError”;
    - Return next(error);
  + });
  + App.use(‘/birds’, function(err, req, res, next)) {
    - If(err.name === “gekkeError”) {
      * Console.log(“geef iets terug”);
    - }
    - Res.status(500).send(err);
  + });
* Geen toegang op req.params.name als deze via een andere functie wordt aangeroepen.
  + Function log(req, res, next) {
    - Console.log(req.params.naam);
    - Next();
  + }
  + App.get(route, log, function);
* Antwoord op mijn vraag: Maak een nieuwe module voor promise wrappers, die ik dan aan app.js en games.js kan meegeven.
  + In een ander uitwerking heb ik dit niet gezien
* Quiz
  + Next() is geen return functie bij zichzelf
  + Bij next met een parameter gaat de next naar een error functie zoeken met 4 parameters om de argument daarin te stoppen.
  + Bij return next wordt de volgende functie niet meer uitgevoerd, zonder de return wel.
  + Als er een next wordt uitgevoerd aan het eind, dan gaat next weer terug naar de vorige functie
  + Zonder next() werkt de functie niet meer
  + Throw new Error. Bij express stuurt deze functie je naar de volgende route, tenzij de functie async is.
  + Volgorde van implementatie is erg belangrijk bij de uitvoer van de functies.
  + ‘/\*’ route verwijst naar alle routes.
  + Een request luistert naar alle paden.
  + Je kunt geen res.json doen nadat je dat al een keer hebt gedaan.