

# Web Programming

## Week 6

*"It's called object-oriented to tell you what you should do with it: object!"*

Phil Wadler, quoted from memory

# Goodie

`Array.from()`

use of optional arguments

# Today: Objects

Basics

Variants of object encoding

Game: OOPSIE

Quiz

# What are Objects?

Data structures +

Methods for access and management

(+ a location for mutable state)

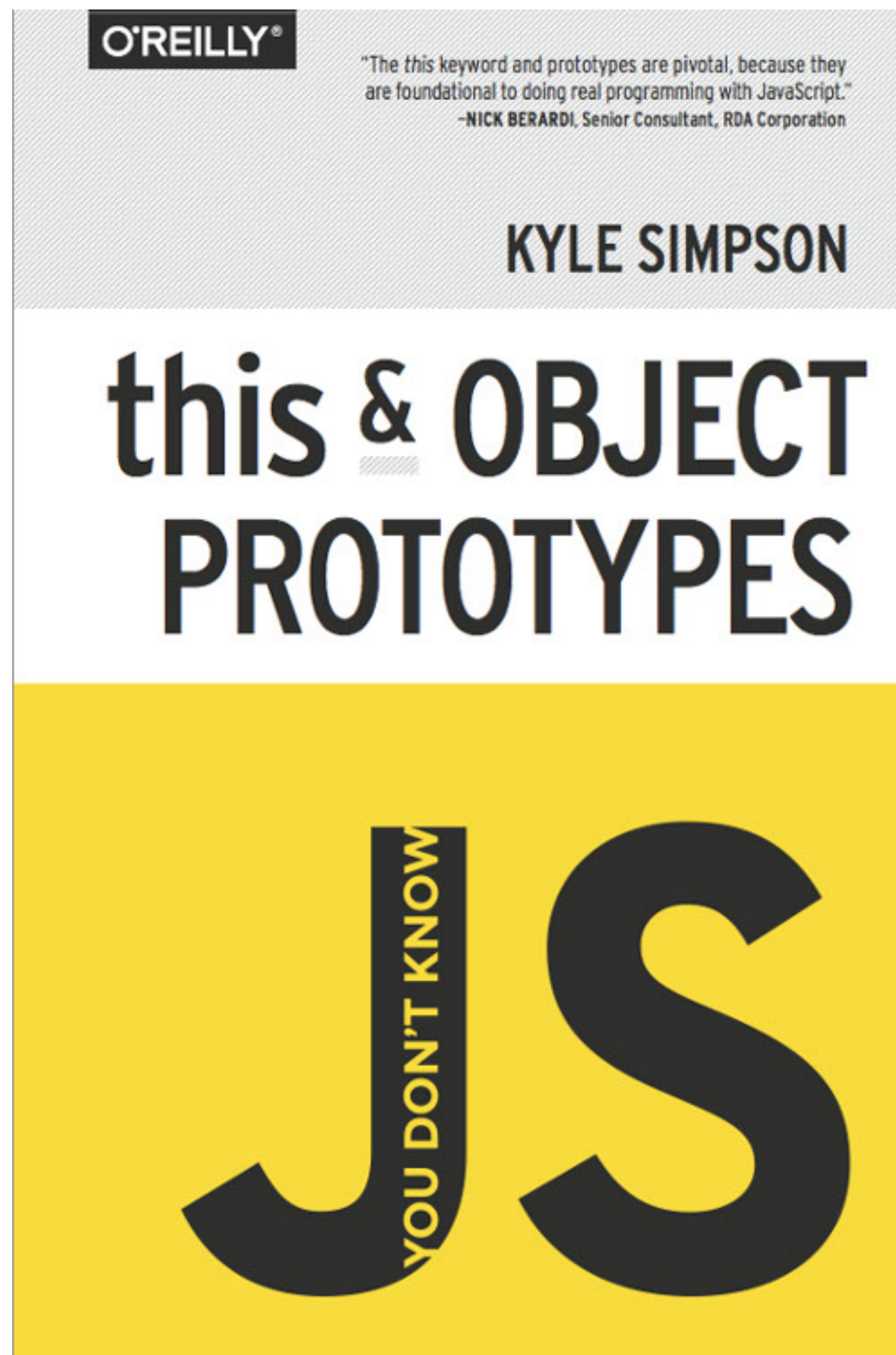
(+ abstraction and polymorphism)

# Different Approaches

Open, dynamic

Closed, explicit

Mixed, classified



# Basics

<https://github.com/getify/You-Dont-Know-JS>

# Open, dynamic

*JS "Objects"*

```
const good = {  
  firstname : "Good",  
  lastname  : "Boy",  
  getName   : function() {  
    return this.firstname + " " + this.lastname  
  }  
};  
// no safety but super dynamic  
// unobvious how to share structure  
// beware of "this"! See Adam Breindl last week.
```

# Closed, explicit

*closure scope, no "this"*

```
function Person(first, last) {  
  let firstname = first; // optional  
  let lastname  = last;  
  return {  
    getName: function() {  
      return firstname + " " + lastname }  
    }  
  }  
}  
// best safety, easy to share structure, but no class
```



# Mixed, classified

*depends on "new"*

```
const Person = ( () => { // lexical scope
  function Person(first, last) { // ctor, binding
    this.firstname = first;
    this.lastname  = last;
  }
  Person.prototype.getName = function() {
    return this.firstname + " " + this.lastname;
  };
  return Person;
}) (); // IIFE
// new Person("Good", "Boy") instanceof Person
```

# Mixed, classified

Is the "default" construction  
vgl. <http://www.typescriptlang.org>

Still dynamic but all "instances" can be  
changed at once by changing the  
prototype!

# Good Parts Reconsidered

- I stopped using `new` years ago.
- I have stopped using `Object.create`.
- I have stopped using `this`.
- I have stopped using `null`.



<https://www.youtube.com/watch?v=DxnYQRuLX7Q>



▶ ⏮ 🔊 32:09 / 1:16:14



# Prototype

Classifies objects similar to a "type"

Manages shared properties

Is itself an object

Can be checked, e.g. by `instanceof`

# Remember: "New"

Creates a **new** Runtime-Scope

Calls the **constructor**-Function  
(cannot be a lambda)

Sets the prototype

# OOPSIE

Throw dice and move forward as often as you want. Throw a 3 and you have to start over "OOPSIE!".

Complete the game with an object construction for the Player (oopsie/oopsie.js) such that the allTests run.

# Fun at Home

Complete OOPSIE for 2 Players with the help of objects.

Extend with a new Rule: you kick out the opponent (back to start) when reaching exactly his field.