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

Object deconstructor

Storybook (initial)

Drehbuch, Intro, Functions
Scientific foundations
Algebraic Data Types
Applied Science, Snake
Scripting, PWA, Plotter, Excel
Objects
Classes
JS Types, JsDoc
Async Programming
Modules
Data Flow, Excel improved
Iterator Protocol, Sequences
Moves, User Interfaces
Crazy JS

Today: Objects

Testing utility (first step)

Variants of object encoding, "this"

Game: OOPSIE

Quiz

What are Objects?

Data structures +

Methods for access and management

(+ a location for mutable state)

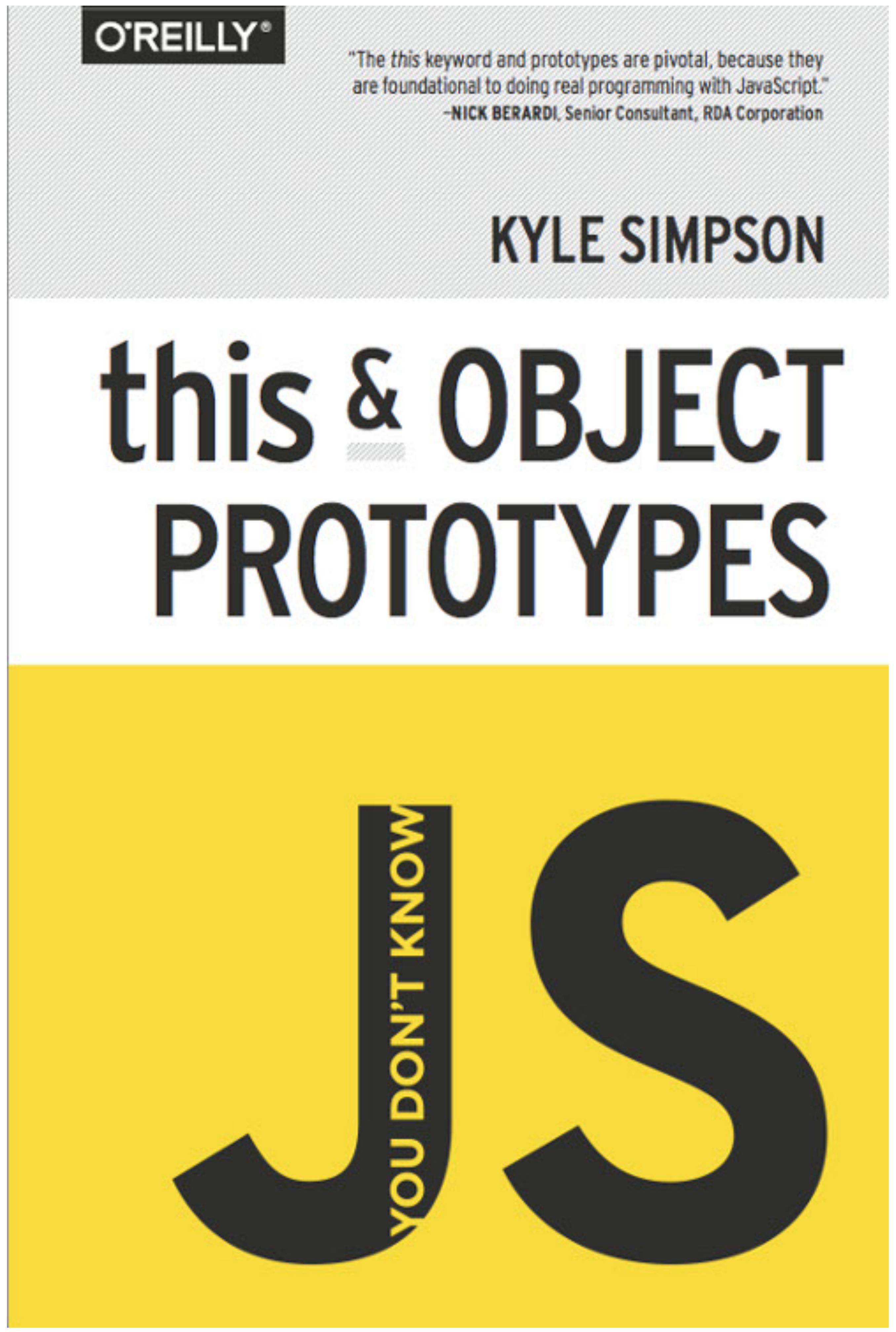
(+ abstraction and polymorphism)

Different Approaches

Open, dynamic

Closed, explicit

Mixed, classified


O'REILLY
"The this keyword and prototypes are pivotal, because they are foundational to doing real programming with JavaScript."
–NICK BERARDI, Senior Consultant, RDA Corporation
KYLE SIMPSON
this & OBJECT PROTOTYPES
YOU DON'T KNOW JS

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) {
    return {
        getName: () => first + " "  + last;
    }
}
// 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. babeljs.io // Node version 4.

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

"this" is an issue

Fundamentally different than Java.

Only "function" delegates "this".

Lambda => has no delegated "this".

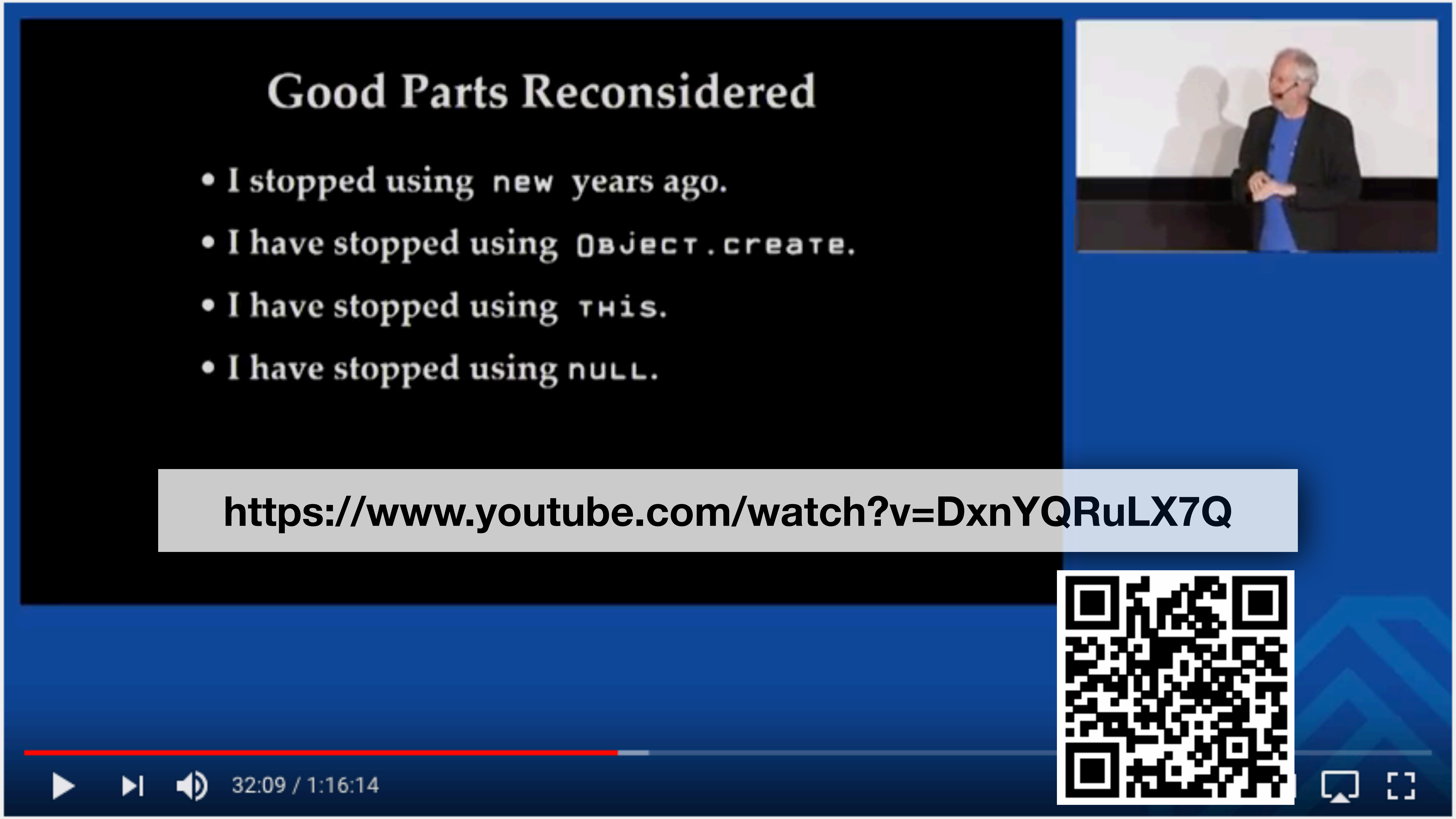
"this" is silly

Receiver
Delegate

```
myObject.foo()
```

rule of thumb

"Calling site: left of the dot."



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

code::dive 2017 – Douglas Crockford – The better parts

Kyle Simpson

Let me put it this way: don't use *this*-aware code unless you really can justify it, and you've carefully weighed the costs. Just because you've seen a lot of code examples slinging around *this* in others' code, doesn't mean that *this* belongs in this code you're writing.

https://github.com/getify/You-Dont-Know-JS/blob/2nd-ed/objects-classes/ch4.md

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.