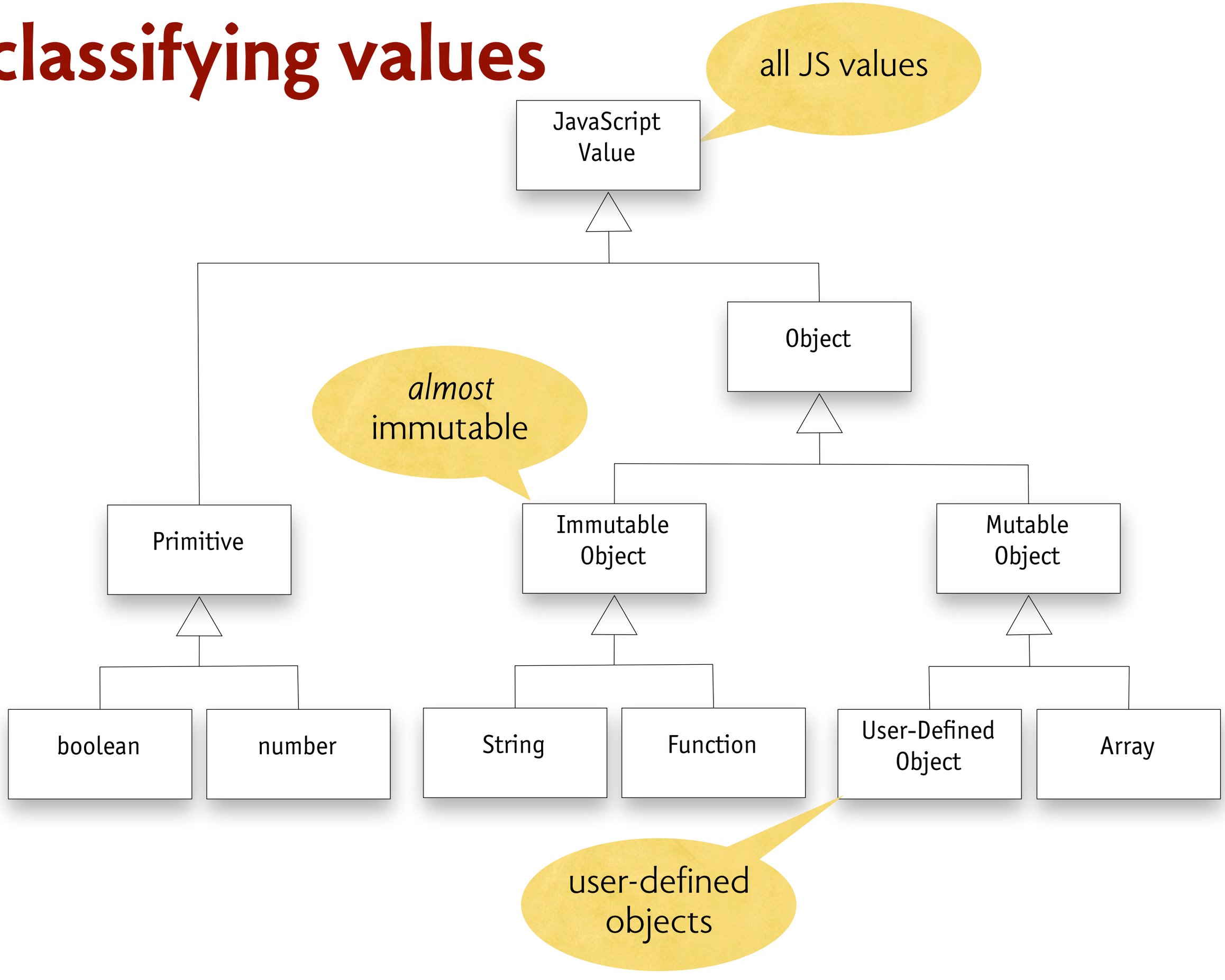
software studio

javascript: values & types

Daniel Jackson

classifying values



typeof

```
> typeof(1)
"number"

> typeof(false)
"boolean"

> typeof('a')
"string"

> typeof([])
"object"

> typeof({})
"object"

> typeof(null)
"object"

> typeof(function () {})
"function"

> typeof(undefined)
"undefined"

> typeof(typeof(1))
"string"
```

note: no array ‘type’

note: no type type

aliasing & immutables

```
> x = ['h', 'i']
["h", "i"]

> y = x
["h", "i"]

> y[1] = 'o'
"o"

> x
["h", "o"]
```

aliasing

```
> x = ['h', 'i']
["h", "i"]

> x[1] = 'o'
"o"

> x
["h", "o"]

> x = 'hi'
"hi"

> x[1]
"i"

> x[1] = 'o'
"o"

> x[1]
"i"
```

some objects
can't be modified

immutables are our friends!

no mention of x
⇒ no change to x

equality and truthiness

like many scripting languages

› features that save 3 seconds of typing
› instead cost 3 hours of debugging

examples

› behavior of built-in equality operator ==
› strange rules of “truthiness”

```
> 0 == ''
  true
> 0 == '0'
  true
> '' == '0'
  false
```

```
> 0 === ''
  false
> 0 === '0'
  false
> '' === '0'
  false
```

```
> (false) ? "true" : "false";
"false"
> (null) ? "true" : "false";
"false"
> (undefined) ? "true" : "false";
"false"
> (0) ? "true" : "false";
"false"
> ('') ? "true" : "false";
"false"
> (undefined == null)
true
> (undefined === null)
false
```

numbers & non-numbers

good news

- no int vs float
- exponents

bad news

- strange NaN value
- no infinite precision

puzzle

- is this good practice?

```
if (f() != NaN) {...}
```

- no, need

```
if (isNaN(f())) {...}
```

```
> 1 + 2
3
> 1.0 + 2
3
> 1/3
0.3333333333333333
> 1000000 * 1000000
1000000000000
> 1000000 * 1000000 * 1000000
1000000000000000000
> 1000000 * 1000000 * 1000000 * 1000000
1e+24
```

```
> 1/0
Infinity
> (1/0) * 0
NaN
```

```
> typeof(NaN)
"number"
> NaN === NaN
false
> NaN !== NaN
true
```

undefined, reference error & null

notions of bad access

› ReferenceError exception
› undefined value (built-in)
› null (predefined value)

how to use them

› ReferenceError: regard as failure if raised
› null: best left unused, IMO
› undefined: unavoidable (eg, optional args)

a little paradox

› undefined is a defined value for a variable

```
> newvar
ReferenceError: newvar is not defined
> newvar = undefined
undefined
> newvar
undefined
> obj = {}
Object
> obj.f
undefined
> obj.f = null
null
```

6.170 Software Studio
Spring 2013