

REGULAR EXPRESSIONS DEEP DIVE

More pattern matching than you ever really
cared to ask about

Can you identify each of these?

- 4246 3151 7734 2234
- 12:34 EDT
- (800) 867-5309
- 457-55-5462
- dwight.k.shrute@dundermifflin.com
- 192.168.0.1
- <http://www.TotsAndNugs.com>
- 11/16/2012
- 16-11-2012

JavaScript can recognize them, too

- It understands a RegExp type.

```
var re = new RegExp(pattern);
```

- or more concisely

```
var re = /pattern/;
```

RegExs are usually tested from the string

- Find all of the matches

```
String.match(RegExp);
```

- Find out where it matches

```
String.search(RegExp);
```

- Replace all occurrences of RegExp with some string

```
String.replace(RegExp, newString);
```

- Split a string using a regex as the delimiter

```
String.split(RegExp);
```

match() returns an array of all matches

```
var addr = document  
    .getElementById('ipaddress').value;  
var numbers = addr.match(/\d+/g);  
for (var x in numbers) {  
    console.log(numbers[x]);  
}
```

search() tells us where the match happens

```
var addr = document
    .getElementById('email').value;
if (addr.search(/^\w+@[a-zA-Z_]+\?\. [a-zA-Z ]
{2,3}$/) === -1)
    throw new Error("Not an email address");
```

- Zero-based answer!!
- -1 means no match

split() allows you to break a string into an array

```
var addr = document  
    .getElementById('address').value;  
var words = addr.split(/\s+/);  
for (var x in words) {  
    console.log(words[x]);  
}
```

replace() replaces regex with something

- It doesn't destroy the original string but returns a copy

```
var email = document.querySelector('#em').value;  
var cleanEmail = email.replace(/@/," at ")  
                      .replace(/\./g," dot ");  
out.innerText = cleanEmail;
```

Modifiers can be placed at the end

- /regex/modifier
- or
- RegExp(regex, modifier)

Modifiers

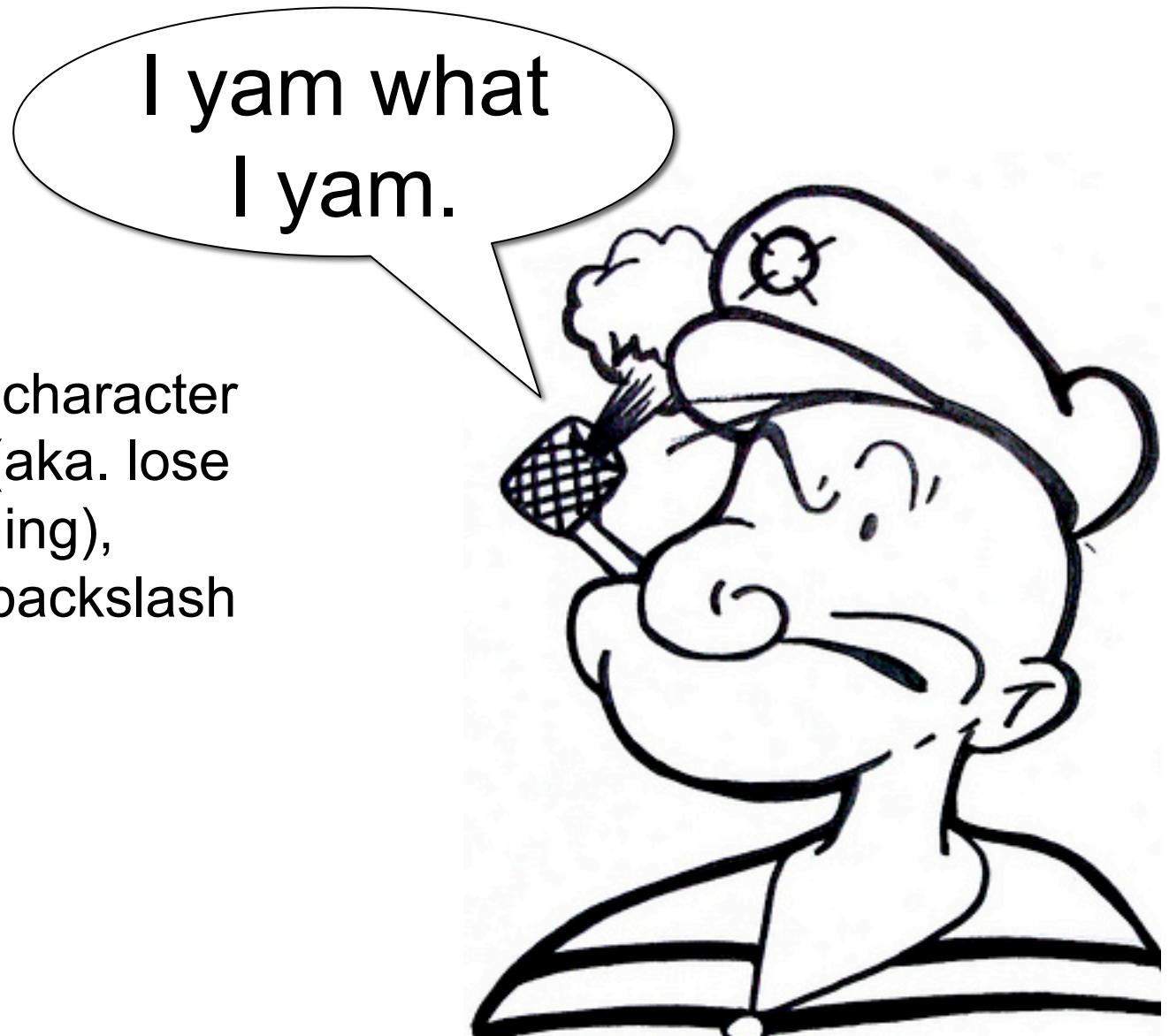
- g means match them all if there are >1 match
- i means ignore case
- m means ignore line breaks

A stylized illustration of a superhero in flight. The superhero has dark hair tied back, a yellow suit with red shoulder accents, and a red cape. They are shown from the waist up, with arms outstretched and a determined expression. The background consists of radiating blue and white lines, suggesting motion and light.

Regex uses metacharacters to
represent types or families of
characters

If it's not a metacharacter, it just is what it is

- Example:
 - a
 - 1
 - T
- To make a metacharacter represent itself (aka. lose its special meaning), precede it by a backslash
- Example:
 - \.
 - \^



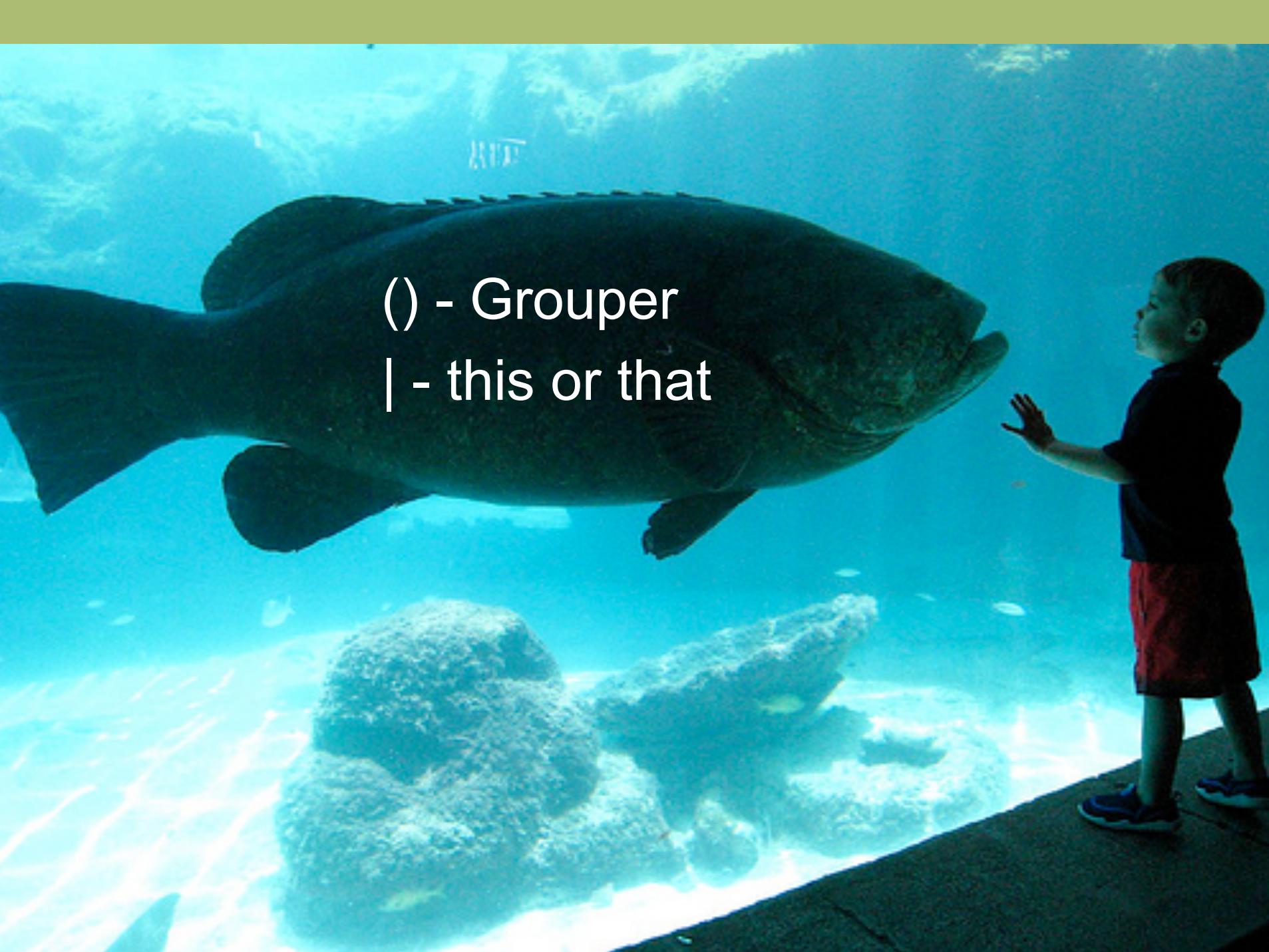
Single character wildcards represent exactly one other character

What it represents	
.	Any character
\d	Any single digit
\D	Any character that is NOT a digit
\w	Any word character (letter, number, underscore)
\W	One character that is NOT a word character
\s	Any single whitespace (space, tab, newline)
\S	One character that is NOT a whitespace
[] • ^ • -	Any one character listed in the brackets • Changes it all to NOT one of these • Ranges of characters



Multipliers turn one character into multiples of one character

- Generic multipliers use curly braces
 - {x}
 - {x,y}
 - {x,}
- Multiplier shortcuts which will save you keystrokes
 - +
 - ?
 - *

A large, dark-colored grouper fish is swimming towards the right side of the frame. A young boy stands on a dark platform to the right, looking up at the fish. He has his left hand raised, palm facing forward, as if reaching out or interacting with the fish. The background is a bright blue, likely the water of a large aquarium tank. The overall scene is a close-up shot with a shallow depth of field.

() - Grouper
| - this or that

Boundaries allow you to match next to something significant

- ^
- \$
- \b
- \B



tl;dr

- Regular Expressions are extremely complex, but extremely powerful
- Fortunately, if you can break them down they're very learnable