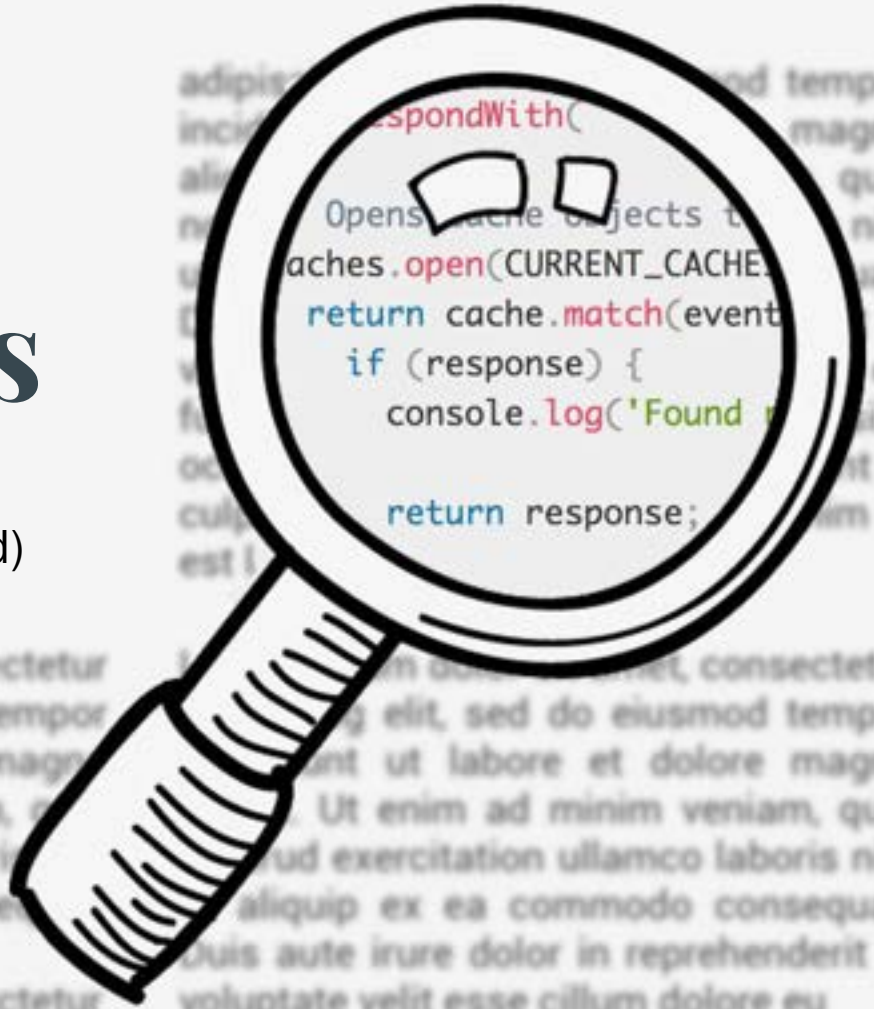
Through The Language Glass

By Chris Nguyen (@uncompiled)



Sapir-Whorf Hypothesis

the structure of a language
affects the speaker's perception
or cognition of the world

Edward Sapir
Benjamin Lee Whorf

Thai Ethnicity and People · Phuket · +6

Why are the Thai people so happy and smiley?

Answer · Request · Follow 1 · Comment · Share · Downvote · ...

4 Answers

Narat Suchartsunthorn, Thai for 24 years
Answered Jan 4, 2015

It's not just people in a service industry, almost every Thais here are smiley. I am one of them and I've never questioned myself why I keep smiling too.

Smile is a common thing to have here. A non-smiley person is often judged as not a friendly person. To be with someone and you don't smile might seem like you don't want to be there or to be with him/her. So we may use them so many ways more than other countries.

เกรงใจ

The Breakdown

เกรง /greng/

fear, be afraid of, be in awe of, dread

ใจ /jai/

mind, heart, spirit.

The Breakdown

เกรง /greng/

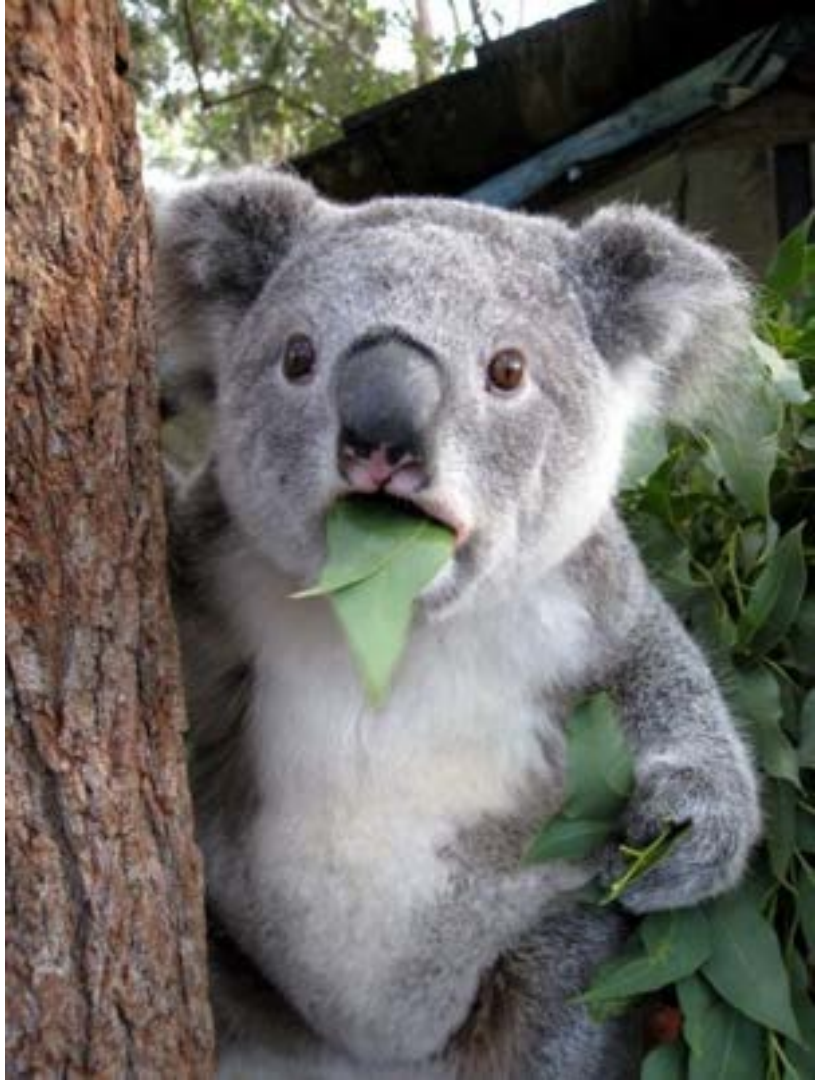
fe

"the discomfort that you feel when you need to inconvenience someone"

ใจ /jai/:

mind, heart, spirit.

blue = green



xanh = xanh

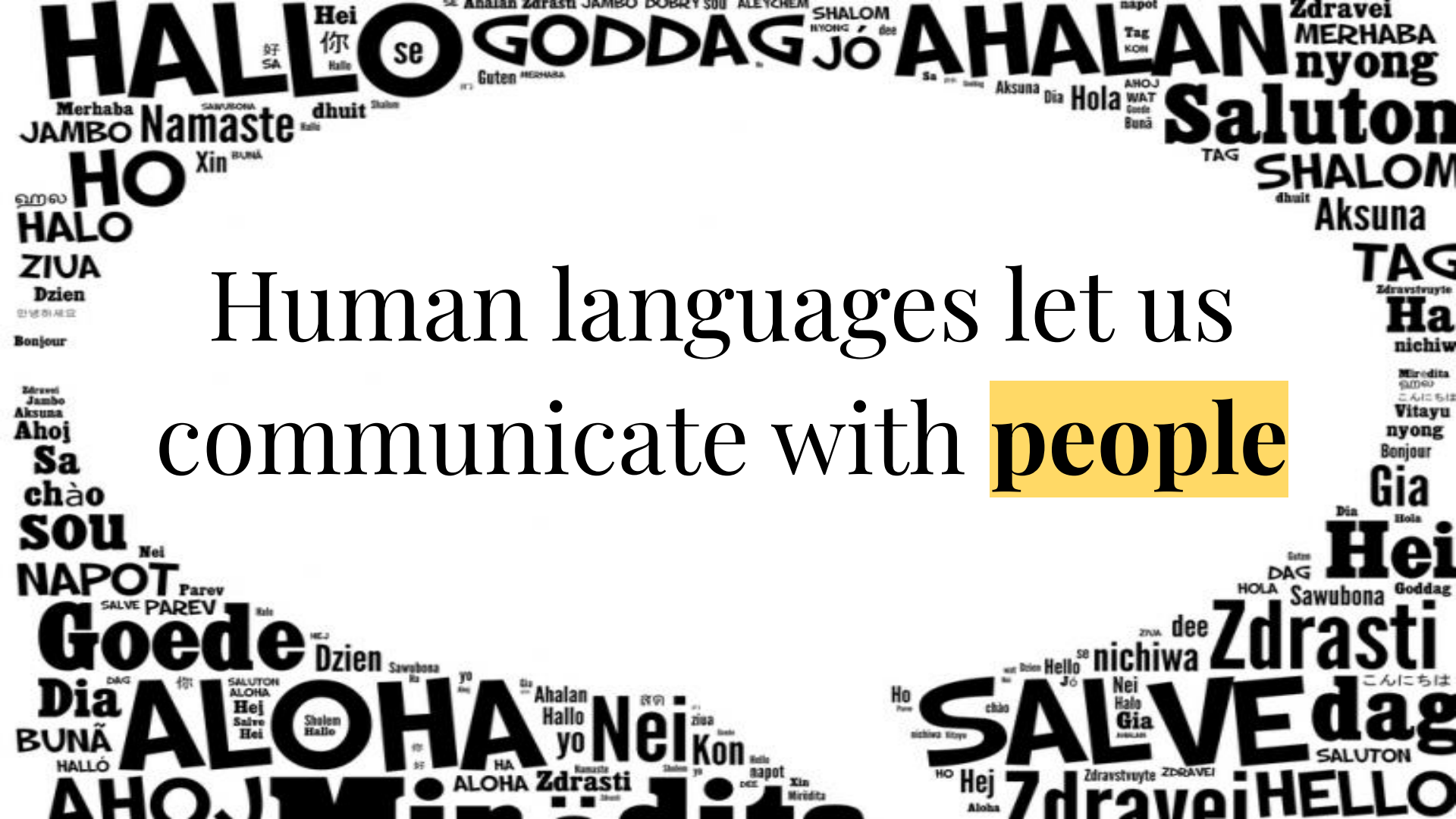
xanh = xanh

xanh = **xanh lá cây**

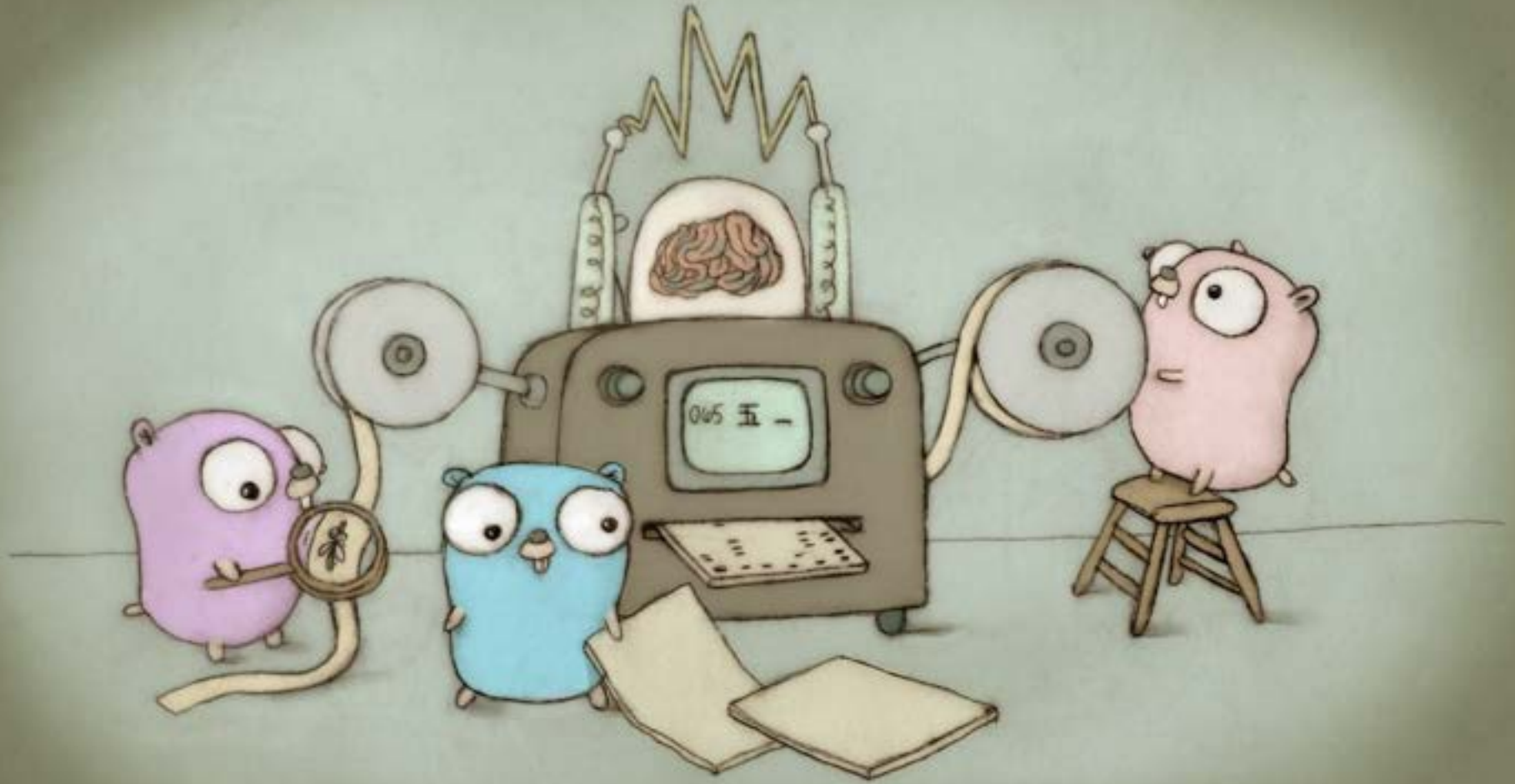
green

Language is a tool for

communication



Human languages let us communicate with **people**


045 五 _


ERLANG

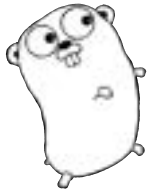

JS


THE
C
PROGRAMMING
LANGUAGE

Imperative vs Declarative

Imperative Languages

- Express **how** to do something
- Similar to how a computer works
- Recipes are real-world examples



Declarative Languages

- Express **what** we want
- Higher-level of abstraction
- Functional languages

HTML is Declarative

```
<img src="cool.gif">
```

VS.

```
createElementinDOM("img")
download("cool.gif")
calculateSize("cool.gif")
drawPixelsOnScreen(...)
```

Most languages are

multi-paradigm

Programming Problem

Given a sequence $s_1, s_2, \ldots s_n$ consisting of **n** lowercase English alphabet characters (a–z), remove all of the characters that occurred previously in the sequence.

Input: [a,b,b,a,c,a,d,a,b,r,a]

Output: [a,b,c,d,r]

Formally: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    let result = [];
    for (let i = 0; i < input.length; i++ ) {
       let curr = input[i];
       if (result.indexOf(curr) < 0)
          result.push(curr);
    }
    return result;
}
```

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    let result = [];
    for (let i = 0; i < input.length; i++) {
        let curr = input[i];
        if (result.indexOf(curr) < 0)
            result.push(curr);
    }
    return result;
}
```

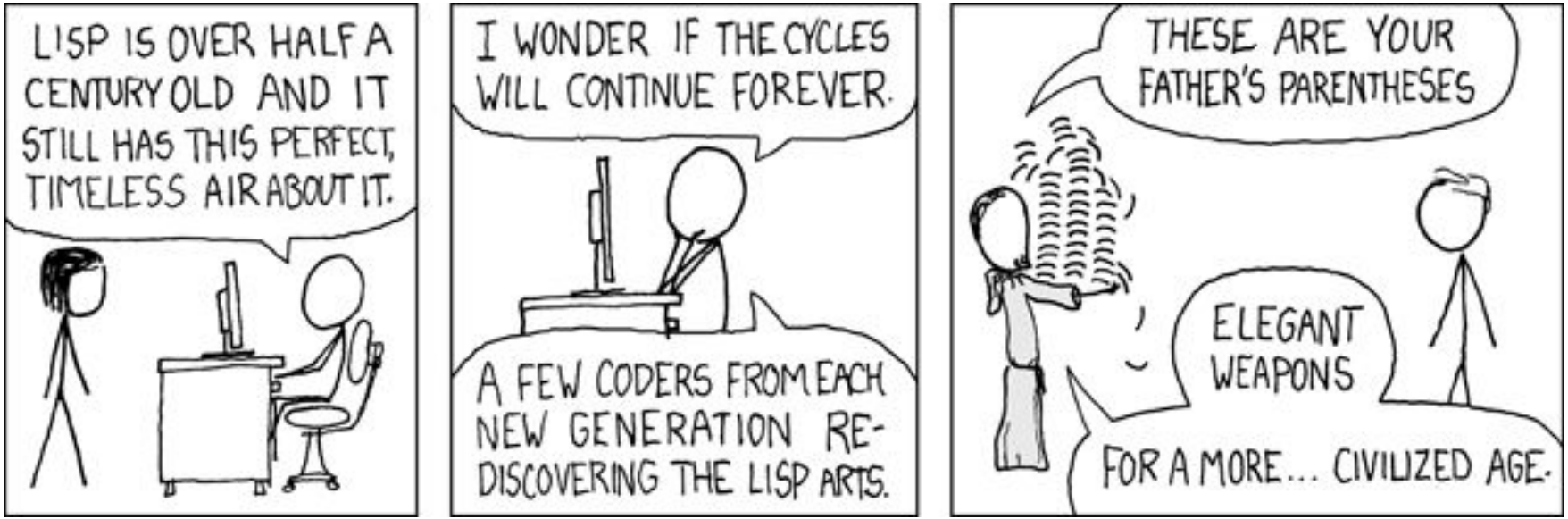
JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    let result = [];
    for (let i = 0; i < input.length; i++) {
        let curr = input[i];
        if (result.indexOf(curr) < 0)
            result.push(curr);
    }
    return result;
}
```

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    let result = [];
    for (let i = 0; i < input.length; i++) {
        let curr = input[i];
        if (result.indexOf(curr) < 0)
            result.push(curr);
    }
    return result;
}
```

Clojure



https://xkcd.com/297/

Clojure: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
(distinct input)
```

Programming languages are **not a zero sum game**

+

higher order functions

=

filter, map, reduce

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    return input.filter(
        (item, index, array) => {
            return index === array.indexOf(item);
        }
    );
}
```

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    return input.filter(
        (item, index, array) => {
            return index === array.indexOf(item);
        }
    );
}
```

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    return input.filter(
        (item, index, array) => {
            return index === array.indexOf(item);
        }
    );
}
```

Set

The **`Set`** object lets you store unique values of any type, whether primitive values or object references.

Syntax

```
new Set([iterable]);
```

Parameters

`iterable`

If an iterable object is passed, all of its elements will be added to the new `Set`. If you don't specify this parameter, or its value is `null`, the new `Set` is empty.

Return value

A new `Set` object.

Description

`Set` objects are collections of values. You can iterate through the elements of a set in insertion order. A value in the `Set` **may only occur once**; it is unique in the `Set`'s collection.

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    return [...new Set(input)];
}
```

JavaScript: Remove all s_i for $\exists j, s_j = s_i$ and $j < i$

```
function solution(input) {
    return Array.from(new Set(input));
}
```

"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

Abraham Maslow

The Psychology of Science, 1966

Languages evolve

DESCRIBED BY urbandictionary.com AS "CARPE DIEM FOR STUPID PEOPLE", YOLO IS ACTUALLY SHORT FOR THIS


STOP TRYING TO MAKE FETCH HAPPEN

MDN web docs
mozilla

Technologies ▾ References & Guides ▾ Feedback ▾

Sign in

Fetch API

Languages Edit

Jump to: Concepts and usage Fetch Interfaces Fetch mixin Specifications Browser compatibility See also

Web technology for developers › Web APIs › Fetch API

The Fetch API provides an interface for fetching resources (including across the network).

Related Topics

Fetch API
- Guides
  - Cross-global fetch usage
  - Fetch basic concepts
  - Using Fetch
- Interfaces
  - Body
  - Headers
  - Request
  - Response
- Methods
  - WindowOrWorkerGlobalScope.fetch()

Yaasssssss!

involved with network requests). This will allow them to be used wherever they are needed in the future, whether it's for service workers, Cache API and other similar things that handle or modify requests and responses, or any kind of use case that might require you to generate your own responses programmatically.

It also provides a definition for related concepts such as CORS and the HTTP origin header semantics, supplanting their separate definitions elsewhere.

For making a request and fetching a resource, use the `GlobalFetch.fetch` method. It is implemented in multiple interfaces, specifically `Window` and `WorkerGlobalScope`. This makes it available in pretty much any context you might want to fetch resources in.

The `fetch()` method takes one mandatory argument, the path to the resource you want to fetch. It returns a `Promise` that resolves to the `Response` to that request, whether it is

Fetch - LS

Usage
Global

% of all users
87.57% + 0.05% = 87.62%

A modern replacement for XMLHttpRequest.

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			64		10.3				
	16	59	65	11	11.2				4
11	17	60	66	11.1	11.3	all	66	11.8	6.2
	18	61	67	TP					
		62	68						
			69						

Notes | Known issues (0) | Resources (7) | Feedback

- Specification [fetch.spec.whatwg.org] ref
- Demo [addyosmani.com] info
- Polyfill [github.com] info
- IE/Edge platform status: Shipped [developer.microsoft.com] official status
- Chrome platform status: Enabled by default [chromestatus.com] official status
- Firefox platform status: 39 [platform-status.mozilla.org] official status
- WebKit platform status: Supported [webkit.org] official status

Evolving the Language

Ecmascript Proposal (TC39)

JDK Enhancement Proposal

Python Enhancement Proposal

github.com/tc39/proposals

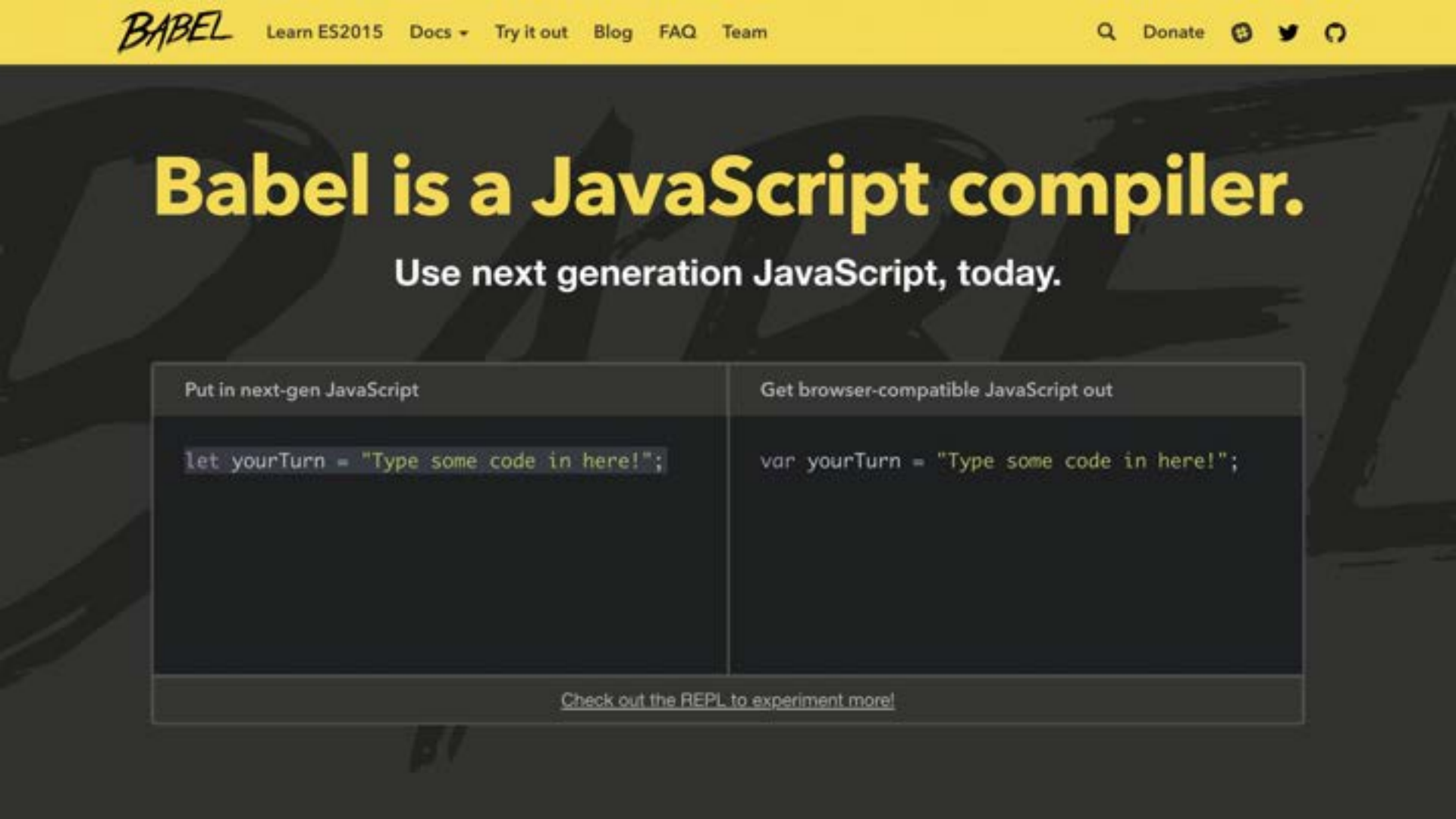
Active proposals

Proposals follow this process document. This list contains only stage 1 proposals and higher that have not yet been withdrawn/rejected, or become finished.

Stage 3

🚀	Proposal	Author	Champion	Tests
🚀	`Function.prototype.toString` revision	Michael Ficarra	Michael Ficarra	✅
	`global`	Jordan Harband	Jordan Harband	✅
	`import()`	Domenic Denicola	Domenic Denicola	❓
	Legacy RegExp features in JavaScript	Claude Pache	Mark Miller Claude Pache	✅
	`BigInt`	Daniel Ehrenberg	Daniel Ehrenberg	🚧

polyfills, **prollyfills**,
and **compilers**


BABEL
Learn ES2015
Docs
Try it out
Blog
FAQ
Team
Donate
Babel is a JavaScript compiler.
Use next generation JavaScript, today.
Put in next-gen JavaScript
Get browser-compatible JavaScript out
let yourTurn = "Type some code in here!";
var yourTurn = "Type some code in here!";
Check out the REPL to experiment more!

github.com/babel/proposals

Babel progress on ECMAScript proposals

Official TC39 Proposals Repo

Proposals

For the official list, check the TC39 repo. This list only contains all proposals higher than Stage 0 that are compilable by Babel. (If Stage 4, it will be in preset-env)

Proposal Link	Current Stage	Status
Rest/Spread Properties	3	3
Asynchronous Iteration	3	3
Dynamic Import	3	3
RegExp Unicode Property Escapes	3	3

github.com/tc39/proposal-pattern-matching

ECMAScript Pattern Matching

Status

Stage: 0

Author: Kat Marchán (npm, @maybekatz)

Champions: Brian Terlson (Microsoft, @bterlson), Sebastian Markbåge (Facebook, @sebmarkbage), Kat Marchán (npm, @maybekatz)

Introduction

This proposal adds a pattern matching expression to the language, based on the existing Destructuring Binding Patterns.

It's structured into multiple parts:

Pattern Matching in Scala

```scala
def int2Word(i: Int): String = i match {
  case 1 => "one"
  case 2 => "two"
  case _ => "many"
}
```

Pattern Matching in Scala

```
def notify(notification: Notification,
           vipList: Seq[String]): String =

  notification match {
    case Email(email, _, _) if vipList.contains(email) =>
      s"You got an email from ${email}!"
    case SMS(number, _) if vipList.contains(number) =>
      s"You got an SMS from ${number}!"
    case _ => processDefaultNotification(other)
  }
```

Pattern Matching in Scala

```scala
def notify(notification: Notification,
           vipList: Seq[String]): String =

  notification match {
    case Email(email, _, _) if vipList.contains(email) =>
      s"You got an email from ${email}!"
    case SMS(number, _) if vipList.contains(number) =>
      s"You got an SMS from ${number}!"
    case _ => processDefaultNotification(other)
  }
```

Pattern Matching in Scala

```
def notify(notification: Notification,
           vipList: Seq[String]): String =

  notification match {
    case Email(email, _, _) if vipList.contains(email) =>
      s"You got an email from ${email}!"
    case SMS(number, _) if vipList.contains(number) =>
      s"You got an SMS from ${number}!"
    case _ => processDefaultNotification(other)
  }
```

Pattern Matching in Scala

```
def notify(notification: Notification,
           vipList: Seq[String]): String =

  notification match {
    case Email(email, _, _) if vipList.contains(email) =>
      s"You got an email from ${email}!"
    case SMS(number, _) if vipList.contains(number) =>
      s"You got an SMS from ${number}!"
    case _ => processDefaultNotification(other)
  }
```

Pattern Matching in Scala

```
def notify(notification: Notification,
           vipList: Seq[String]): String =

  notification match {
    case Email(email, _, _) if vipList.contains(email) =>
      s"You got an email from ${email}!"
    case SMS(number, _) if vipList.contains(number) =>
      s"You got an SMS from ${number}!"
    case _ => processDefaultNotification(other)
  }
```

Pattern Matching in JS

```
const res = await fetch(jsonService)
const val = match (res) {
  {status: 200, headers: {'Content-Length': s}} =>
    `size is ${s}`,
  {status: 404} =>
    'JSON not found',
  {status} if (status >= 400) =>
    throw new RequestError(res)
}
```

Pattern Matching in JS

```
const res = await fetch(jsonService)
const val = match (res) {
  {status: 200, headers: {'Content-Length': s}} =>
    `size is ${s}`,
  {status: 404} =>
    'JSON not found',
  {status} if (status >= 400) =>
    throw new RequestError(res)
}
```

Pattern Matching in JS

```
const res = await fetch(jsonService)
const val = match (res) {
  {status: 200, headers: {'Content-Length': s}} =>
    `size is ${s}`,
  {status: 404} =>
    'JSON not found',
  {status} if (status >= 400) =>
    throw new RequestError(res)
}
```

github.com/zkat/pattycake

Install

```
$ npm install pattycake
```

Table of Contents

Example

```
import match, {$} from 'pattycake'

const res = await fetch(jsonService)
const val = match (res) (
  {
    status: 200,
    headers: {'Content-Length': $}
```

Through the Language Glass

- Languages are communities
- Languages are not a zero sum game
- New languages open your mind to alternative ways of thinking
- Concepts stick with you regardless of the language that you're currently using

Chris Nguyen
@uncompiled

Thanks!