

明日には使えなくなる
ES7トーク

自己紹介

azu

@azu_re

Web scratch,
JSer.info

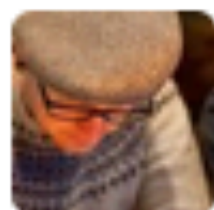


ES7 Proposals

[tc39/ecma262](https://github.com/tc39/ecma262)

用語

- TC : Technical Committee = 専門委員会
- TC39: ECMAScriptを策定してる専門委員会
- プロポーサル : 仕様の提案
- ECMAScript: JavaScriptの仕様
- ES7: ECMAScript7



Mikeal

@mikeal



Follow

@domenic any documentation for this “even TC39 is moving to a feature-based model.” ?

↩ Reply ↻ Retweet ★ Favorite ⋮ More

1:25 AM - 12 Oct 2014



Domenic Denicola @domenic · Oct 12

@mikeal @wycats has been working on a post, but this is all public, see e.g. github.com/tc39/ecma262

↩ Reply ↻ Retweet ★ Favorite ⋮ More



TC39 is moving to a feature-based model

- ECMAScript 7は機能ごとに仕様を策定していく(方針)
- 仕様同士を独立して進めていく事でスピードをあげる
- 仕様のモジュール化
- [tc39/ecma262](https://tc39.es/ecma262)に現在のプロポーサルが一覧がある
- それぞれの仕様はTC39のプロセスで策定が進められる

TC39のプロセスとは

- Stage 0. Strawman : ESに入りたいアイデアを議論する段階
- Stage 1. Proposal : Strawmanを具体化、デモ作成、分析
- Stage 2. Draft : 正式な仕様定義の形式で仕様書を書く段階
- Stage 3. Candidate: Draftの実装等をしてフィードバック
- Stage 4. Finished: ECMAScriptに正式採用 - Test262への実

¹ 参考 [JS] ECMAScript6をまるっと学ぶ。重要用語とか、仕様策定の進め方とか、新機能とか。 - YoheiM.NET

今日話すのは**Stage : 0**の 話が中心


明日なくなるかもしれない仕様の話

後あまり正確ではないです

Object.observe ★

Stage 2

Object.observe()

- JavaScriptオブジェクトの変更を監視する仕様
-  Google Chrome 36には既に載ってる
- Object.observe() でデータバインディング革命 - HTML5 Rocks

// データを持ったモデル

```
var model = {};
```

// modelを監視する

```
Object.observe(model, function(changes){
```

```
    changes.forEach(function(change) {
```

```
        // 変更内容
```

```
        console.log(change.type, change.name, change.oldValue);
```

```
    });
```

```
});
```

// modelを変更する

```
model.some = "追加";
```

Exponentiation Operator ★ Stage 2

- `**` 演算子の仕様
- べき乗演算子
- `x ** y == Math.pow(x, y)` のこと
- 9月のTC39 MTGでStage 2まで上がった
- 5.8 Exponentiation Operator Update

Exponentiation Operator

- Python, CoffeeScript, F#, Ruby, Perl 等他の言語にもある
- 仕様自体は`Math.pow`そのままなのでシンプル
- Traceurに実装済み
- [Add support for the exponentiation operator by arv · Pull Request #1216 · google/traceur-compiler](#)

Async Functions ★

Stage 1

- `async`と`await`の仕様
- [Task.js](#):`spawn`のシンタックスシュガー的な感じ
- Generator関数とPromiseを使った同期的な非同期処理
- [regenerator](#)にTranspileの実装がある
- [Transform async functions and await expressions by benjamn · Pull Request #101 · facebook/regenerator](#)

Array.prototype.contains ★ Stage 1

Array.prototype.contains のモチベーション

```
if (arr.indexOf(e1) !== -1) {  
    ...  
}
```

というコードを無くして次のようにしたい

```
if (arr.contains(e1)){  
    ...  
}
```

BREAK THE WEBの問題

- [1075059 - non-enumerable Array.prototype.contains is not web-compatible \(breaks jsfiddle.net\)](#)
- [gist:28953b01e455078fb4f8](#)
- [Array.prototype.contains solutions](#)

Types and Type Annotation ★ Stage 0

tc39-notes/sept-25.md

Types and Type Annotations

- JSDocやTypeScript等色々な型注釈が溢れてる
- 短期目標
 - まずはその構文を予約しておきたい
 - その構文を使った場合はSyntax Errorに落とす

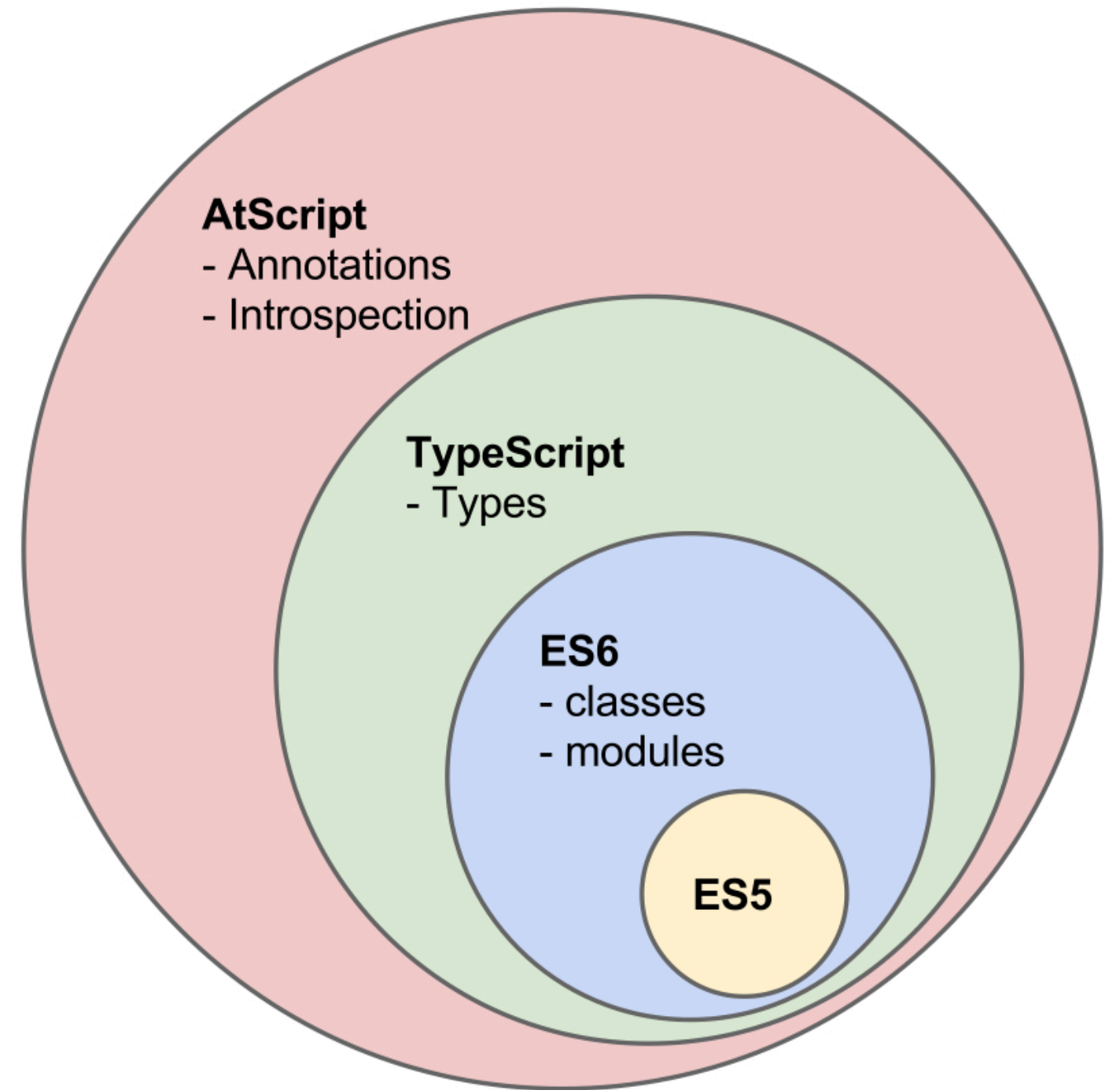
Types and Type Annotations

- 長期目標
 - その構文でType Annotationsの実装、型チェック
 - d.tsのようなAPIのドキュメント定義に使いたい
- 類似研究
 - TypeScript
 - Python

AtScript ★ Stage NaN

AtScript

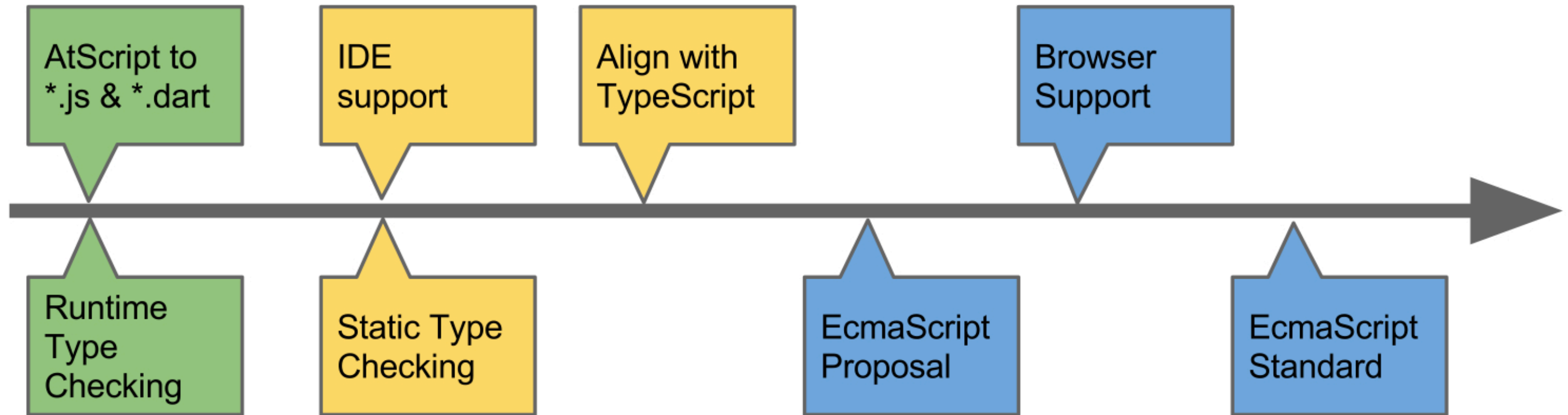
- Angular 2.0で使われている
- ES6+A(Annotations)
- TraceurでES6 validに変換できる
 - [traceur-compiler 入門](#)
- まずは[Assert.js](#)を使ったruntime assertから



AtScript is ES6+A(Annotations)

- Type Annotations
 - TypeScriptでやるやつ
- Metadata Annotations
 - メタデータを定義する`@Directive`
- Introspection
 - DIなどで**実行時**に使えるメタ情報の提供

AtScript Roadmap



AtScript Resources

- [Keynote: AtScript](#)
- [ES6 & Traceur](#)
- [AtScript Primer](#)
- [AtScript \(was "ES6 +A"\) Q&A](#)

Flow (Facebook) ★ Stage NaN

- [@Scale 2014: Recap of Web Track | Engineering Blog | Facebook Code](#)
- Facebook社内のType assertions 静的チェックツール?
- [React](#)に使われている

Flow(Facebook)

- Compatible TypeScript Syntax
- Committed to evolved with JS standard
- Integrated with React + JSX

-- youtu.be/M8x0bc81smU?t=12m47s

ECMAScriptと型

- AtScriptとFlow(Facebook) どちらもTypeScriptのSyntaxをベースに置いている
- TypeScriptのチームとAtScript/Flowのチームは話し合ってる

The TypeScript team is working with both the Flow and AtScript teams to help ensure that resources

– <http://blogs.msdn.com/b/typescript/archive/2014/10/22/typescript-and-the-road-to-2-0.aspx>

global.asap ★ Stage 0

- ES6でJob Queuesというキューの仕組みが入った
- ES6 Promises等で利用している
- `global.asap` はそのキューに優先度を付けたいという話

```
// high order queue  
global.asap(function()  
  
});
```


Trailing Commas in Function Call Expressions and Declarations ★ Stage

1

[jeffmo/es-trailing-function-commas](https://jeffmo.github.io/es-trailing-function-commas)

関数呼び出しと定義のケツカンマ問題

```
1: function clownsEverywhere(  
2:   param1,  
3:   param2,  
4:   param3,  
5: ) { /* ... */ }  
6:  
7: clownsEverywhere(  
8:   'foo',  
9:   'bar',  
10:  'baz',  
11: );
```

関数呼び出しと定義のケツカンマ問題

- 現在の仕様だと関数定義の引数の末尾に, があるとエラー
- 同様に関数呼び出しの末尾に, があってもエラー
- これを許容したいという提案
- **モチベーション**
 - gitなどでのdiffのわかり易さ
 - コード生成のし易さ

Call Expressions

CallExpression :

MemberExpression Arguments

CallExpression Arguments

CallExpression [Expression]

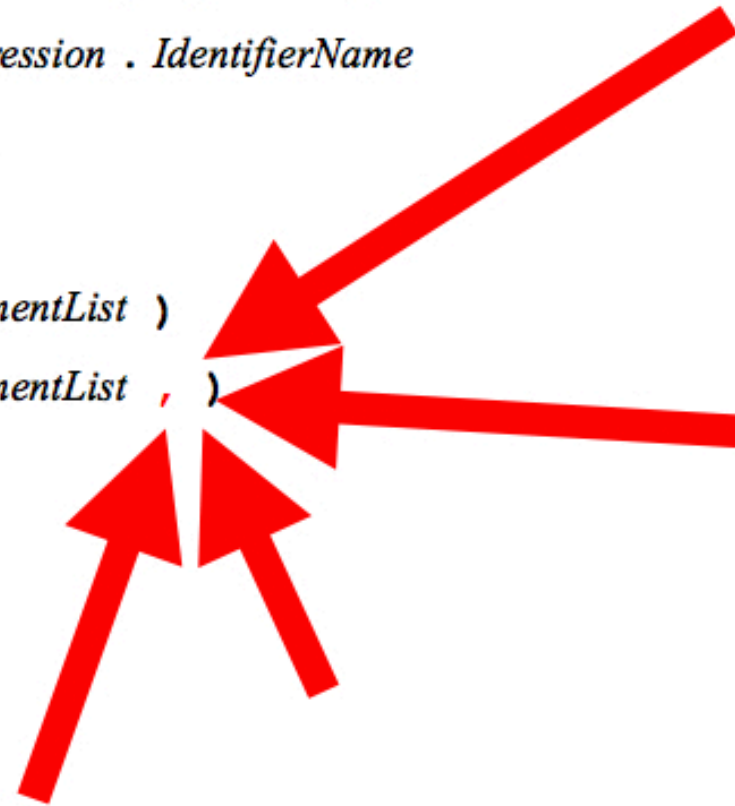
CallExpression . IdentifierName

Arguments :


()

(ArgumentList)

(ArgumentList ,)



おわりに

- rwaldron/tc39-notes にTC39のMTGノートがまとまっている
 - Follow  [@rwaldron](https://twitter.com/rwaldron)
- ポッドキャスト聞く感覚で読むと面白いと思います。
 - TC39 MTGのMTGをしたい
- 次回のMTGで話す事はtc39/agendasにまとめられています