

Typed Ember

Strong Types for Better Apps

James C. Davis



@jamescdavis



@jamescdavis



@jamescdavis

About Me



About Me

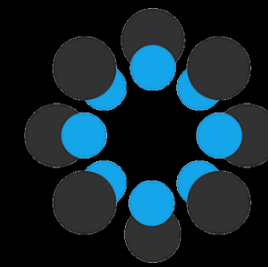
About My Work

About My Work


- Front-end Lead, Center for Open Science

About My Work


- Front-end Lead, Center for Open Science
- Open Science Framework (osf.io)




About My Work

- Front-end Lead, Center for Open Science
- Open Science Framework (osf.io) 
- github.com/CenterForOpenScience/ember-osf-web


About My Work

- Front-end Lead, Center for Open Science
- Open Science Framework (osf.io) 
 - github.com/CenterForOpenScience/ember-osf-web
- Typed-Ember Team


About My Work

- Front-end Lead, Center for Open Science
- Open Science Framework (osf.io) 
 - github.com/CenterForOpenScience/ember-osf-web
- Typed-Ember Team
 - ember-cli-typescript

About My Work

- Front-end Lead, Center for Open Science
- Open Science Framework (osf.io) 
 - github.com/CenterForOpenScience/ember-osf-web
- Typed-Ember Team
 - ember-cli-typescript
 - Ember and Ember Data type definitions

About My Work

- Front-end Lead, Center for Open Science
- Open Science Framework (osf.io) 
 - github.com/CenterForOpenScience/ember-osf-web
- Typed-Ember Team
 - ember-cli-typescript
 - Ember and Ember Data type definitions
 - Type definitions for popular addons

The image shows the TypeScript logo, which consists of a blue square with the letters 'TS' in white. The 'T' is a simple, bold, sans-serif letter. The 'S' is also a bold, sans-serif letter, slightly more stylized with a rounded top and bottom. The letters are centered within the blue square.

TS



¿por qué?

“Automatic” Documentation



```
/**
 * @param {string} name
 * @param {Date} date
 * @param {bool} isCool
 * @return {bool}
 */
function canDoCoolStuff(name, date, isCool) {

    return name === 'James' && date > Date() && isCool;
}
```



```
/**
 * @param {string} name
 * @param {Date} date
 * @param {bool} isCool
 * @return {bool}
 */
function canDoCoolStuff(name, date, isCool) {

    return name === 'James' && date > Date() && isCool;
}
```



```
function canDoCoolStuff(name: string, date: Date, isCool: boolean): boolean {

    return name === 'James' && date > Date() && isCool;
}
```



```
/**
 * @param {string} name
 * @param {Date} date
 * @param {bool} isCool
 * @return {bool}
 */
function canDoCoolStuff(name, date, isCool) {

    return name === 'James' && date > Date() && isCool;
}
```



```
function canDoCoolStuff(name: string, date: Date, isCool: boolean) {

    return name === 'James' && date > Date() && isCool;
}
```

Reduce Run-time Errors



```
function foo(arg) {  
  return arg.bar;  
}
```




```
function foo(arg) {  
    return arg.bar;  
}
```

```
foo( );
```



```
function foo(arg) {  
  return arg.bar;  
}
```

```
foo( );
```

TypeError: Cannot read property 'bar' of undefined



```
function foo(arg) {  
  return arg.bar;  
}
```

```
foo( );
```

TypeError: Cannot read property 'bar' of undefined





```
function foo(arg) {  
    return arg.baz( );  
}
```



```
function foo(arg) {  
  return arg.baz( );  
}
```

```
foo({ baz: true });
```



```
function foo(arg) {  
  return arg.baz();  
}
```

```
foo({ baz: true });
```

TypeError: arg.baz is not a function



```
function foo(arg) {  
  return arg.baz();  
}
```

```
foo({ baz: true });
```

TypeError: arg.baz is not a function

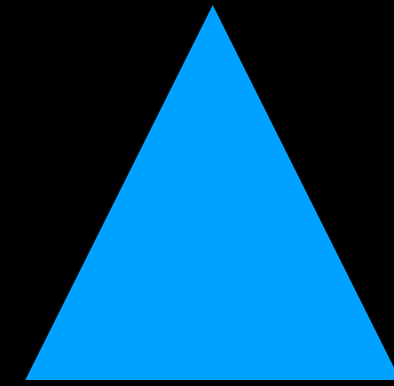
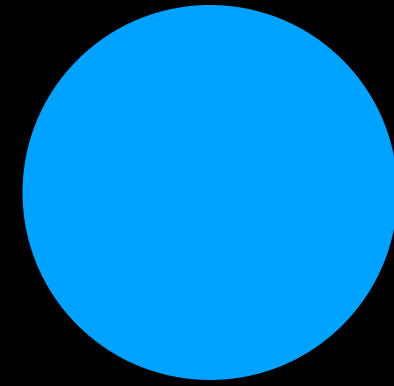
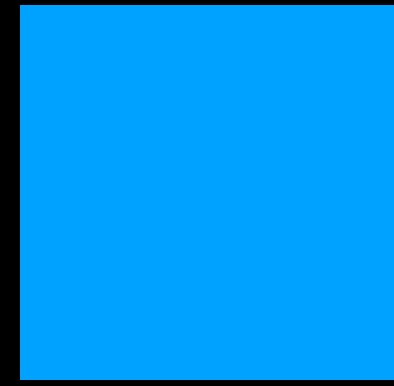


TypeError

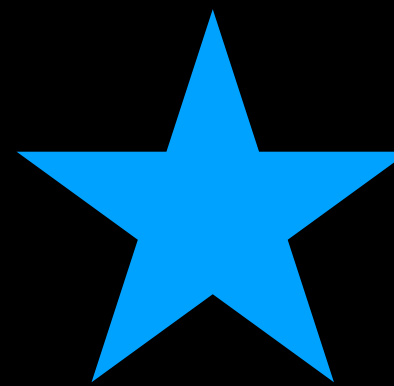
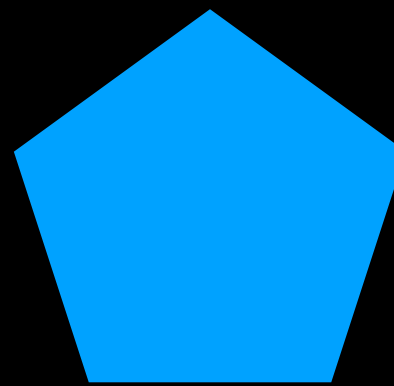
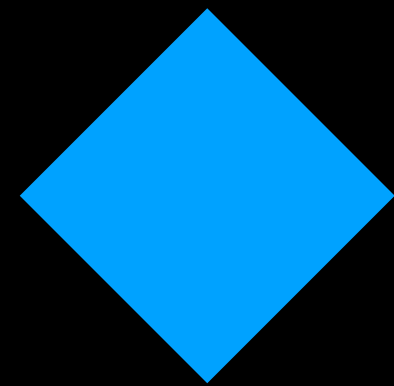


Refactor with wild abandon

What's a type?



What's a type?





```
type Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}
```



```
type Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}  
  
let speaker: Person = {  
  name: 'James'  
};
```



```
interface Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}  
  
let speaker: Person = {  
  name: 'James'  
};
```



```
type Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}
```

```
let speaker: Person = {  
  name: 'James',  
  height: 'tall',  
  birthdate: Date(),  
  isMarried: true,  
};
```




```
type Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}  
  
let speaker: Person = {  
  name: 'James',  
  height: 'tall',  
  birthdate: Date(),  
  isMarried: true,  
};
```



```
type Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}
```

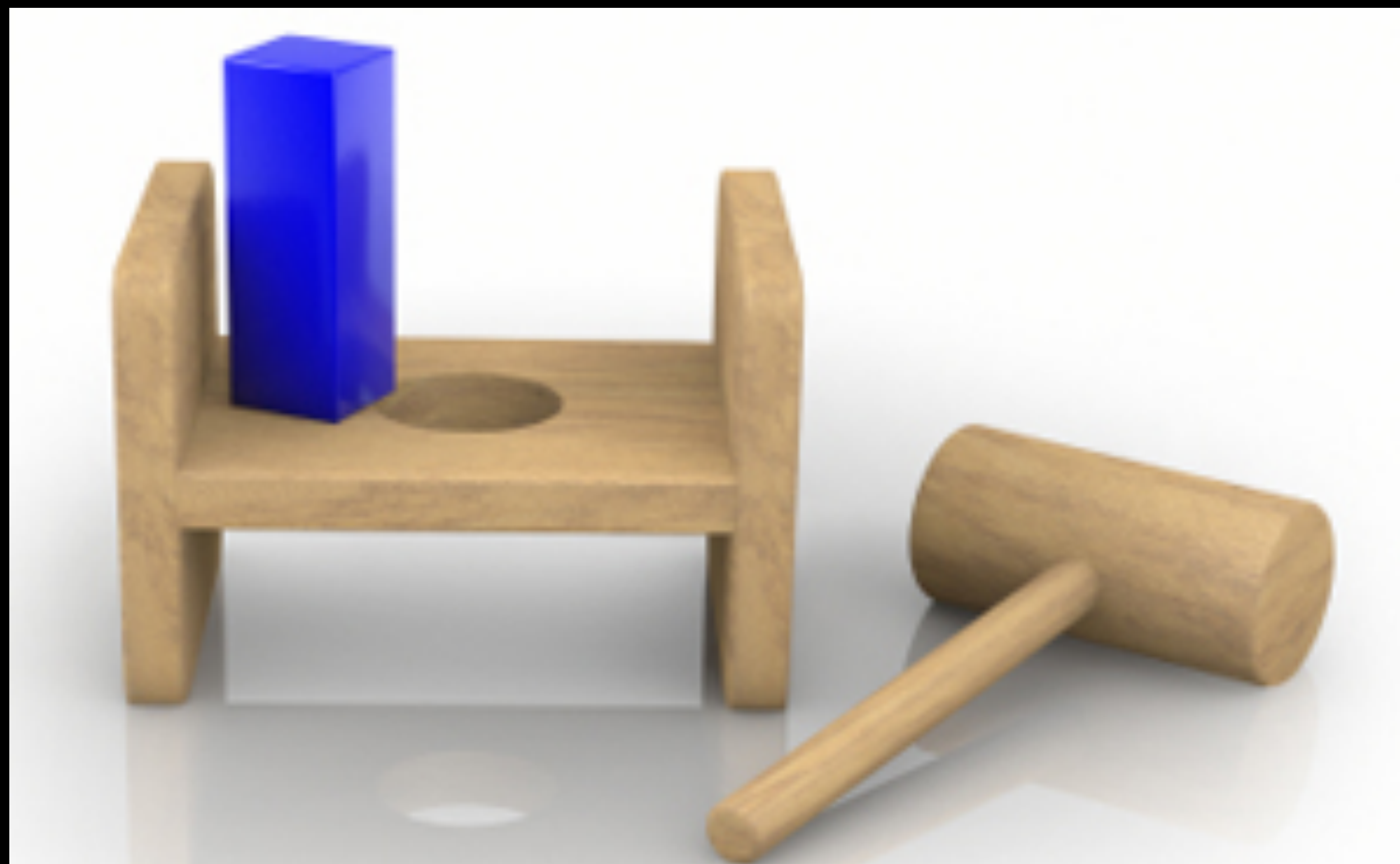
```
let speaker: Person = {  
  name: 'James',  
  height: 123,  
  birthdate: Date(),  
  isMarried: true,  
};
```



```
type Person = {  
  name: string;  
  height: number;  
  birthdate: Date;  
  isMarried: boolean;  
}
```

```
let speaker: Person = {  
  name: 'James',  
  height: 123,  
  birthdate: Date(),  
  isMarried: true,  
};
```





How do we TypeScript in Ember?

ember-cli-typescript

build-time type checking

ember-cli-typescript

build-time type checking

compilation to Javascript

ember-cli-typescript

build-time type checking

compilation to Javascript

ember-cli-typescript

type definitions

build-time type checking

compilation to Javascript

ember-cli-typescript

type definitions

blueprints

How does it look like?

Welcome!

We hope you find exactly what you're looking for in a place to stay.

[About Us](#)[View Larger](#)

Grand Old Mansion

Owner: *Veruca Salt*

Type: *Standalone - Estate*

Location: *San Francisco*

Number of bedrooms: *15*



```
// app/models/rental.js

import DS from 'ember-data';

const { Model, attr } = DS;

export default class Rental extends Model {
  @attr() title;
  @attr() owner;
  @attr() city;
  @attr() category;
  @attr() image;
  @attr() bedrooms;
  @attr() description;
}
```



```
// app/models/rental.ts

import DS from 'ember-data';

const { Model, attr } = DS;

export default class Rental extends Model {
  @attr() title: string;
  @attr() owner: string;
  @attr() city: string;
  @attr() category: string;
  @attr() image: string;
  @attr() bedrooms: number;
  @attr() description: string;
}
```

```
// app/services/map-element.js
```

```
import { camelize } from '@ember/string';
import { set } from '@ember/object';
import Service from '@ember/service';
import { inject as service } from '@ember/service';

export default class MapElement extends Service {
  @service geocode;
  @service map;

  constructor() {
    if (!this.cachedMaps) {
      set(this, 'cachedMaps', {});
    }
    super(...arguments);
  }

  async getMapElement(location) {
    let camelizedLocation = camelize(location);
    let element = this.cachedMaps[camelizedLocation];
    if (!element) {
      element = this._createMapElement();
      let geocodedLocation = await this.geocode.fetchCoordinates(location);
      this.map.createMap(element, geocodedLocation);
      this.cachedMaps[camelizedLocation] = element;
    }
    return element;
  }

  _createMapElement() {
    let element = document.createElement('div');
    element.className = 'map';
    return element;
  }
}
```

```
// app/services/map-element.ts
```

```
import { camelize } from '@ember/string';
import { set } from '@ember/object';
import Service from '@ember/service';
import { inject as service } from '@ember/service';

export default class MapElement extends Service {
  @service geocode: Geocode;
  @service map: Map;

  cachedMaps: { [loc: string]: HTMLElement };

  constructor() {
    if (!this.cachedMaps) {
      set(this, 'cachedMaps', {});
    }
    super(...arguments);
  }

  async getMapElement(location: string) {
    let camelizedLocation = camelize(location);
    let element = this.cachedMaps[camelizedLocation];
    if (!element) {
      element = this._createMapElement();
      let geocodedLocation = await this.geocode.fetchCoordinates(location);
      this.map.createMap(element, geocodedLocation);
      this.cachedMaps[camelizedLocation] = element;
    }
    return element;
  }

  _createMapElement() {
    let element = document.createElement('div');
    element.className = 'map';
    return element;
  }
}
```



```
<ListFilter @filter={{action "filterByCity"}} as |filteredResults|>
  <ul class="results">
    {{#each filteredResults as |rentalUnit|}}
      <li><RentalListing @rental={{rentalUnit}} /></li>
    {{/each}}
  </ul>
</ListFilter>
```



```
import { action } from '@ember/object';
import Component from '@glimmer/component';
import { tracked } from '@glimmer/tracking';
import Rental from 'super-rentals/models/rental';

export default class RentalListing extends Component {
  @tracked isWide = false;
  args: { rental: Rental };

  @action
  toggleImageSize() {
    this.isWide = !this.isWide;
    if (this.args.rental) {
      alert(this.args.rental.bedrooms * 2);
    }
  }
}
```




```
import { action } from '@ember/object';
import Component from '@glimmer/component';
import { tracked } from '@glimmer/tracking';
import Rental from 'super-rentals/models/rental';

export default class RentalListing extends Component {
  @tracked isWide = false;
  args: { rental: Rental };

  @action
  toggleImageSize() {
    this.isWide = !this.isWide;
    if (this.args.rental) {
      alert(this.args.rental.bedroom * 2);
    }
  }
}
```



Dang! Looks like a Type Error.

```
app/components/rental-listing.ts:19:34 - error TS2551: Property 'bedroom' does not exist on type 'Rental'.  
Did you mean 'bedrooms'?
```

```
19      alert(this.args.rental.bedroom * 2);  
                                ~~~~~
```

```
app/models/rental.ts:12:11  
12  @attr() bedrooms!: number;  
      ~~~~~  
'bedrooms' is declared here.
```

Environment

typescript@3.3.3333, ember-cli-typescript@2.0.0



```
import { action } from '@ember/object';
import Component from '@glimmer/component';
import { tracked } from '@glimmer/tracking';
import Rental from 'super-rentals/models/rental';

export default class RentalListing extends Component {
  @tracked isWide = false;
  args: { rental: Rental };

  @action
  toggleImageSize() {
    this.isWide = !this.isWide;
    if (this.args.rental) {
      alert(this.args.rental.bedrooms * 2);
    }
  }
}
```



```
import { action } from '@ember/object';
import Component from '@glimmer/component';
import { tracked } from '@glimmer/tracking';
import Rental from 'super-rentals/models/rental';

export default class RentalListing extends Component {
  @tracked isWide = false;
  args: { rental: Rental };

  @action
  toggleImageSize() {
    this.isWide = !this.isWide;
    if (this.args.rental) {
      alert(this.args.rental.city * 2);
    }
  }
}
```



Dang! Looks like a Type Error.

```
app/components/rental-listing.ts:19:17 - error TS2362: The left-hand side of an arithmetic operation must be  
of type 'any', 'number', 'bigint' or an enum type.
```

```
19      alert(this.args.rental.city * 2);  
      ~~~~~
```

Environment

typescript@3.3.3333, ember-cli-typescript@2.0.0

FAQs

Do I have to convert
everything at once?

Do I have to convert
everything at once?

Nope!

Where do I start?

models

Where do I start?

models

services

Where do I start?

models

services

Where do I start?

shared components

But what about addons?

precompile to JS

But what about addons?

precompile to JS

include type definitions

But what about addons?

precompile to JS

include type definitions

But what about addons?

TS benefits for everybody!

Pain Points

Pain Points

- New syntax and concepts to learn

Pain Points

- New syntax and concepts to learn
- Type definitions for third-party addons

Pain Points

- New syntax and concepts to learn
- Type definitions for third-party addons
 - Definitely Typed `@types/*`


Pain Points

- New syntax and concepts to learn
- Type definitions for third-party addons
 - Definitely Typed `@types/*`
- Type definitions for Ember and Ember Data

Pain Points

- New syntax and concepts to learn
- Type definitions for third-party addons
 - Definitely Typed `@types/*`
- Type definitions for Ember and Ember Data
 - also Definitely Typed `@types/ember` `@types/ember-data`

Learn More

- Repo: github.com/typed-ember/ember-cli-typescript
- Docs: typed-ember.github.io/ember-cli-typescript
- Help: #e-typescript 

Thank you!

