

3

<template> tag

packages

@ember/component	Input, Textarea
@ember/helper	array, concat, fn, get, hash, uniqueld
@ember/modifier	on
@ember/routing	LinkTo
@ember/template	htmlSafe
@embroider/util	ensureSafeComponent

3_a

components

my-addon: src/components/hello.hbs

```
1 <div
2   class={{local
3     this.styles
4     "container"
5     (if this.someCondition (array "hide" "after-3-sec"))
6   }}
7 >
8   {{t "hello.message" name=@name}}
9 </div>
```

my-addon: src/components/hello.ts

```
1 import Component from '@glimmer/component';
2
3 import styles from './hello.css';
4
5 export default class Hello extends Component<HelloSignature> {
6   styles = styles;
7
8   get someCondition(): boolean {
9     return true;
10  }
11 }
```


● ● ● my-addon: src/components/hello.ts

```
1  import Component from '@glimmer/component';
2
3  import styles from './hello.css';
4
5  export default class Hello extends Component<HelloSignature> {
6    styles = styles;
7
8    get someCondition(): boolean {
9      return true;
10   }
11 }
```

● ● ● my-addon: src/components/hello.gts

```
1  import Component from '@glimmer/component';
2
3  import styles from './hello.css';
4
5  export default class Hello extends Component<HelloSignature> {
6    styles = styles;
7
8    get someCondition(): boolean {
9      return true;
10   }
11
12   <template>
13   </template>
14 }
```

● ● ● my-addon: src/components/hello.gts

```
1  import Component from '@glimmer/component';
2
3  import styles from './hello.css';
4
5  export default class Hello extends Component<HelloSignature> {
6    styles = styles;
7
8    get someCondition(): boolean {
9      return true;
10   }
11
12   <template>
13     <div
14       class={{local
15         this.styles
16         "container"
17         (if this.someCondition (array "hide" "after-3-sec"))
18       }}
19     >
20       {{t "hello.message" name=@name}}
21     </div>
22   </template>
23 }
```

● ● ● my-addon: src/components/hello.gts

```
1  import { array } from '@ember/helper';
2  import Component from '@glimmer/component';
3  import { t } from 'ember-intl';
4  import { local } from 'embroider-css-modules';
5
6  import styles from './hello.css';
7
8  export default class Hello extends Component<HelloSignature> {
9    styles = styles;
10
11    get someCondition(): boolean {
12      return true;
13    }
14
15    <template>
16      <div
17        class={{local
18          this.styles
19          "container"
20          (if this.someCondition (array "hide" "after-3-sec"))
21        }}
22      >
23        {{t "hello.message" name=@name}}
24      </div>
25    </template>
26  }
```

● ● ● my-addon: src/components/hello.gts

```
1 import { array } from '@ember/helper';
2 import Component from '@glimmer/component';
3 import { t } from 'ember-intl';
4 import { local } from 'embroider-css-modules';
5
6 import styles from './hello.css';
7
8 export default class Hello extends Component<HelloSignature> {
9   get someCondition(): boolean {
10     return true;
11   }
12
13   <template>
14     <div
15       class={{local
16         styles
17         "container"
18         (if this.someCondition (array "hide" "after-3-sec"))
19       }}
20     >
21       {{t "hello.message" name=@name}}
22     </div>
23   </template>
24 }
```

● ● ● my-addon: src/components/hello.gts

```
1  import type { TOC } from '@ember/component/template-only';
2  import { t } from 'ember-intl';
3
4  import styles from './hello.css';
5
6  const Hello: TOC<HelloSignature> = <template>
7    <div class={{styles.container}}>
8      {{t "hello.message" name=@name}}
9    </div>
10 </template>;
11
12 export default Hello;
```

● ● ● my-addon: src/components/hello.gts

```
1 import type { TOC } from '@ember/component/template-only';
2 import { t } from 'ember-intl';
3
4 import styles from './hello.css';
5
6 <template>
7   <div class={{styles.container}}>
8     {{t "hello.message" name=@name}}
9   </div>
10 </template> satisfies TOC<HelloSignature>;
```


● ● ● my-addon: src/index.ts

```
1 export { default as Hello } from './components/hello.gts';
```

● ● ● my-addon: src/template-registry.ts

```
1 import type HelloComponent from './components/hello.gts';  
2  
3 export default interface MyAddonRegistry {  
4   Hello: typeof HelloComponent;  
5   hello: typeof HelloComponent;  
6 }
```


3_b

routes

● ● ● my-app: app/templates/index.hbs

```
1 <div class={{this.styles.container}}>
2   <Hello @name={{this.userName}} />
3 </div>
```

● ● ● my-app: app/templates/index.gts

```
1 <template>
2   <div class={{this.styles.container}}>
3     <Hello @name={{this.userName}} />
4   </div>
5 </template>
```

● ● ● my-app: app/templates/index.gts

```
1 import { Hello } from 'my-addon';
2
3 <template>
4   <div class={{@controller.styles.container}}>
5     <Hello @name={{@controller.userName}} />
6   </div>
7 </template>
```

● ● ● my-app: app/templates/index.gts

```
1 import type { TOC } from '@ember/component/template-only';
2 import { Hello } from 'my-addon';
3 import type IndexController from 'my-app/controllers/index';
4
5 interface IndexSignature {
6   Args: {
7     controller: IndexController;
8     model: unknown;
9   };
10 }
11
12 <template>
13   <div class={{@controller.styles.container}}>
14     <Hello @name={{@controller.userName}} />
15   </div>
16 </template> satisfies TOC<IndexSignature>;
```

● ● ● my-app: app/templates/index.gts

```
1  import type { TOC } from '@ember/component/template-only';
2  import { Hello } from 'my-addon';
3  import type IndexController from 'my-app/controllers/index';
4
5  import styles from './index.css';
6
7  interface IndexSignature {
8    Args: {
9      controller: IndexController;
10     model: unknown;
11   };
12 }
13
14 <template>
15   <div class={{styles.container}}>
16     <Hello @name={{@controller.userName}} />
17   </div>
18 </template> satisfies TOC<IndexSignature>;
```

3_c

tests

● ● ● my-app: tests/integration/components/hello-test.ts

```
1  import {
2    render,
3    type TestContext as BaseTestContext,
4  } from '@ember/test-helpers';
5  import { hbs } from 'ember-cli-htmlbars';
6  import { setupRenderingTest } from 'my-app/tests/helpers';
7  import { module, test } from 'qunit';
8
9  interface TestContext extends BaseTestContext {
10    userName: string;
11  }
12
13  module('Integration | Component | hello', function (hooks) {
14    setupRenderingTest(hooks);
15
16    hooks.beforeEach(function (this: TestContext) {
17      this.userName = 'Zoey';
18    });
19
20    test('it renders', async function (this: TestContext, assert) {
21      await render<TestContext>(
22        hbs`
23          <Hello @name={{this.userName}} />
24        `,
25      );
26
27      assert.dom().hasText('Hello, Zoey!');
28    });
29  });
```


● ● ● my-app: tests/integration/components/hello-test.gts

```
1 import {
2   render,
3   type TestContext as BaseTestContext,
4 } from '@ember/test-helpers';
5 import { Hello } from 'my-addon';
6 import { setupRenderingTest } from 'my-app/tests/helpers';
7 import { module, test } from 'qunit';
8
9 interface TestContext extends BaseTestContext {
10   userName: string;
11 }
12
13 module('Integration | Component | hello', function (hooks) {
14   setupRenderingTest(hooks);
15
16   hooks.beforeEach(function (this: TestContext) {
17     this.userName = 'Zoey';
18   });
19
20   test('it renders', async function (this: TestContext, assert) {
21     await render(
22       <template>
23         <Hello @name={{this.userName}} />
24       </template>,
25     );
26
27     assert.dom().hasText('Hello, Zoey!');
28   });
29 });
```

● ● ● my-app: tests/integration/components/hello-test.gts

```
1  import {
2    render,
3    type TestContext as BaseTestContext,
4  } from '@ember/test-helpers';
5  import { Hello } from 'my-addon';
6  import { setupRenderingTest } from 'my-app/tests/helpers';
7  import { module, test } from 'qunit';
8
9  interface TestContext extends BaseTestContext {
10    userName: string;
11  }
12
13  module('Integration | Component | hello', function (hooks) {
14    setupRenderingTest(hooks);
15
16    hooks.beforeEach(function (this: TestContext) {
17      this.userName = 'Zoey';
18    });
19
20    test('it renders', async function (this: TestContext, assert) {
21      const self = this;
22
23      await render(
24        <template>
25          <Hello @name={{self.userName}} />
26        </template>,
27      );
28
29      assert.dom().hasText('Hello, Zoey!');
30    });
31  });
```

● ● ● my-app: tests/integration/components/hello-test.gts

```
1  import {
2    render,
3    type TestContext as BaseTestContext,
4  } from '@ember/test-helpers';
5  import { Hello } from 'my-addon';
6  import { setupRenderingTest } from 'my-app/tests/helpers';
7  import { module, test } from 'qunit';
8
9  interface TestContext extends BaseTestContext {
10    userName: string;
11  }
12
13  module('Integration | Component | hello', function (hooks) {
14    setupRenderingTest(hooks);
15
16    hooks.beforeEach(function (this: TestContext) {
17      this.userName = 'Zoey';
18    });
19
20    test('it renders', async function (this: TestContext, assert) {
21      const { userName } = this;
22
23      await render(
24        <template>
25          <Hello @name={{userName}} />
26        </template>,
27      );
28
29      assert.dom().hasText('Hello, Zoey!');
30    });
31  });
```

● ● ● my-app: tests/integration/components/hello-test.gts

```
1 import { render } from '@ember/test-helpers';
2 import { Hello } from 'my-addon';
3 import { setupRenderingTest } from 'my-app/tests/helpers';
4 import { module, test } from 'qunit';
5
6 module('Integration | Component | hello', function (hooks) {
7   setupRenderingTest(hooks);
8
9   const userName = 'Zoey';
10
11   test('it renders', async function (assert) {
12     await render(
13       <template>
14         <Hello @name={{userName}} />
15       </template>,
16     );
17
18     assert.dom().hasText('Hello, Zoey!');
19   });
20 });
```

● ● ● my-app: tests/integration/components/hello-test.gts

```
1 import { render } from '@ember/test-helpers';
2 import { Hello } from 'my-addon';
3 import { setupRenderingTest } from 'my-app/tests/helpers';
4 import { module, test } from 'qunit';
5
6 module('Integration | Component | hello', function (hooks) {
7   setupRenderingTest(hooks);
8
9   test('it renders', async function (assert) {
10     const userName = 'Zoey';
11
12     await render(
13       <template>
14         <Hello @name={{userName}} />
15       </template>,
16     );
17
18     assert.dom().hasText('Hello, Zoey!');
19   });
20 });
```