EMBER

DATA DOWN ACTION UP(DDAU)

Core concepts:

1.**route**

Routes are handled by route handler

-load templte

-load model

-reidrect to a diff route

-actions 🡪 changing model or route

Appln route -> wen application first olad up

Index route – default for every /

this.route('index', { path: '/' });

**Nested routes**

the user navigates to /post/5, the route will then have the post\_id of 5 to use to load the correct post. Ember follows the convention of :model-name\_id for two reasons. The first reason is that Routes know how to fetch the right model by default, if you follow the convention. The second is that params is an object, and can only have one value

**Route handler**

Model()->The route will then set the return value from the model hook as the model property of the controller. You will then be able to access the controller's model property in your template:

**Dynamic Models – routes with dynamic segments**

, it's important that we include some information in the URL about not only which template to display, but also which model.

In Ember, this is accomplished by defining routes with [dynamic segments](https://guides.emberjs.com/v2.14.0/routing/defining-your-routes/#toc_dynamic-segments).

*Once you have defined a route with a dynamic segment, Ember will extract the value of the dynamic segment from the URL for you and pas s them as a hash to the model hook as the first argument:*

*send data using student.performacen object.id -> from url(routter.js variable name) ->passed to model*

multiple model - > RSVP HASH

paramsFor -> parameter of parentroute / Using paramsFor will also give you the query params defined on that route's controller

modelFor

controllerFor

This method could also be used to look up the current route's parameters from an action or another method on the route, and in that case we have a shortcut: this.paramsFor(this.routeName).

**Route – default name templae**

**Route\_name.js**

**->templateName property**

**Redirection:**

transitionTo() behaves exactly like the [link-to](https://guides.emberjs.com/v2.14.0/templates/links) helper.

If the new route has dynamic segments, you need to pass either a model or an identifier for each segment. Passing a model will skip the route's model() hook since the model is already loaded.

Model(params){}

Beforemodel(/transition/)

afterModel(model,trannsiton)

redirect(model.transition) ->no parent fire

**PREVENTING AND RETRYING**

PREvent from parent - > willTransition ->linkto trantionTO

Child -> beforemodel model aftermodel -> trans.abort()

Save transations in routes controller and reuse it retry() in controller

**Loading -state**

When accessing foo.bar route then Ember will search for:

1. foo.bar-loading
2. foo.loading or foo-loading
3. loading or application-loading

It's important to note that foo.bar.loading is not considered now.

Loading –event

Actions:

{

Loading:function(transiotn,originroute)

{

}

}

**Error substate** similar to loading

Foo.bar-error

Error event

Error:function(error,transition)

**Query parameter**

/articles?category=recent

queryParams: ['category'],

category: null,

this ll cause the URL to update the query param. By default, a query param property change won't cause a full router transition (i.e. it won't call model hooks and setupController, etc.); it will only update the URL. ->oso everything should be done in computed propertied of controller

**link-to query params**

{{#link-to students {query-params mark=’math’}}} mark shd be defined in students controller /anything in hierarchy

**transitionTO query params**

this.transitionTo(‘students’,model,{queryParams:{subject:’math’}}) **or** this.transitionTo('/posts/1?sort=date&showDetails=true');

**default value –wont be serialized , name of pareter in controller need not be same as queryparameter name**

***queryParams :{***

***category :’subject’***

***}***

*Category : ‘math’*

**Sticky query param**

change a query param, it is stored and tied to the model loaded into the route.

**2.2Promises**

The router considers any object with a then() method defined on it to be a promise.

Object which gives eventual result

2.**templates** –helpers –handlebar templating language

-*if unless each with*

*{{#if}}*

*{{/if}}*

{{ }} for helpers, components , varaibles

{{rental-list rental=rentalUnit}}

Helpers are most likely used to manipulate data further before presenting to the users

3.**components**

4.**Helpers**

**{{rupees\_to\_dollars}}**

Helpers are most likely used to manipulate data further before presenting to the users

-Ember g helper rupees-to-dollars

-rupees-to-dolalrs.js

Function rupeesToDollars([params])

{

}

. A helper is usually a simple function that can be used in any template

2 ways

1.iniline invocation {{if isFast "zoooom" "putt-putt-putt"}}

Returns zoom or

2.block invocation Use block helpers to render only part of a template.

3.nested invocation {{if isFast (isTrue "zoooom" “no”)}}

Inline - > returns a single value

**5.Services and Utilities**

Utiltities -> reusable code

5 **model**

Models define the type of data that will be provided by your server.

Ember data

store as a read-through cache for your app's models.

is a library that integrates tightly with Ember to make it easy to retrieve models from your server as JSON, save updates back to the server, and create new models in the browser.

|  |  |
| --- | --- |
| Router.map(function() {  this.route('photo', { path: '/photos/:photo\_id' });  }); | |
| app/routes/photo.js | |
| 1  2  3  4  5  6  7 | import Ember from 'ember';  export default Ember.Route.extend({  model(params) {  return this.get('store').findRecord('photo', params.photo\_id);  }  }); |

Both your components and routes have access to this shared store; when they need to display or modify a model, they first ask the store for it..

## Adapter

An **adapter** is an object that translates requests from Ember (such as "find the user with an ID of 1") into requests to a server

### Transforms

Transforms to the datatype(Deserialization)

DS.attr(‘bool’) -> t,true,1,”1” => Boolean true

Query single recorg

This.get(‘store’).findRecord(type,id)

This.get(‘store’).findRecord(‘article’,’grandold-mansion’)

[] notation will not work--you'll have to use objectAt(index) instead.

Query ->queryparams

. Calling [store.query()](http://emberjs.com/api/data/classes/DS.Store.html" \l "method_query) will make a GET request with the passed object serialized as query params.

|  |  |
| --- | --- |
| 4  5  6  7  8 | // GET to /persons?filter[name]=Peter  this.get('store').query('person', {  filter: {  name: 'Peter'  }  }).then(function(peters) {  // Do something with `peters`  }); |

6.addons

7.modelFor()

{{Outlet}} ->helper it could be used to load the nested template

8.Components

Reusable html inside a block

Template, .js

Component 🡪action dasherized

-shd atleast have one dash, so it doesn’t interfere

**Component subclass** -1.action

2.change component element

3.change element dynamically inside component

**Dynamic rendering** {{component}} helper

Component lifecycle

didReceiveattr /didupdatearttr-> better than observer,wen attr changes

didi

**passing properties -> can be internal properties also, defined in subclass of compoentn**

{{component-name title=post.tile body=body}}

-either params , named

**{{yield}} 🡪 component used as block form**

Template inside the block ll be replaced to yield

|  |  |
| --- | --- |
| 2  3  4 | {{#blog-post title=title}}  <p class="author">by {{author}}</p>  {{body}}  {{/blog-post}} |

*It's important to note that the template scope inside the component block is the same as outside. If a property is available in the template outside the component, it is also available inside the component block.*

**Contextual compoennets**

{{yield}} 🡪 {{body}}

{{yield (hash body=(component bstyle))}}

*Components built this way are commonly referred to as "Contextual Components", allowing inner components to be wrapped within the context of outer components without breaking encapsulation.*

# **Customizing a Component's Element** [**Edit Page**](https://github.com/emberjs/guides/edit/master/source/localizable/components/customizing-a-components-element.md)

tagName,

classNames:[],

**classNameBindings**:className => div class=’abc’

className :’abc’

Boolean

classNameBindings:isTrue:enable:disable

isTrue :false

**attributebinding**

**{{component-name a b c as |aa bb cc|}}**

*A b c ->component params,positionalparams*

*Aa bb cc->yielded block params used by block*

*hsblock*

**Componetn actions**

**Eventhandlers:**

Click,

onMouseOver

eg)

click(event)

{}

Or {{action methodName}}

<p {{action callMe}} /P> 🡪 actions:{

callMe()

{

}

}

<p onclick={{action callMe}} /P> 🡪 actions:{

callMe(event)

{

}

}

*To utilize an event object as a function parameter:*

* *Define the event handler in the component (which is designed to receive the browser event object).*
* *Or, assign an action to an inline event handler in the template (which creates a closure action and does receive the event object as an argument).*

***In Ember, each component can have a property called actions, where you put functions that can be*** [***invoked by the user interacting with the component itself***](https://guides.emberjs.com/v2.14.0/templates/actions/)***, or by child components.***

***Action ‘ ’ -> see action local to the component***

***Action “ ”***

*Note that deleteCurrentUser is no longer in quotes here as opposed to* [*previously*](https://guides.emberjs.com/v2.14.0/components/triggering-changes-with-actions/#toc_passing-the-action-to-the-component)*. Quotes are used to initially pass the action down the component tree, but at every subsequent level you are instead passing the actual function reference (without quotes) in the action helper.*

## Controllers

Controllers behave like a specialized type of Component that is rendered by the router when entering a Route.

sendAction

Components are like black box to maintain reusablility.

Component shd expose some methods, for it to be used by the route/component in which it is called.

*Subcomponent used in maincompoent like*

*<h2>{{gas-and-electricity/select-fuel-type componentName=componentName toggleEdit='callInParentComponent'}}</h2>*

Whne we clikc a p tag, toggle the property

click()

{

//this.get('toggleEdit')();

this.sendAction('toggleEdit');

or

this.sendAction('toggleEdit',this.componentFullName);

},

Parent component or controller

actions:

{

callInParentComponent(name)

{

alert("callInParentComponent.."+name)

}

}

This.attrs.dontEdit is equivalent to this.get(‘dontEdit’)

Value –don’t need to define in component.

<button type="submit" onclick={{action 'sendValue' inputVal}}>submit</button>

GIT

GIT INIT -> creates git file

Three stages

Modified staged comiiited

Git add -> adds files for next commit

GIT ADD FILE/FOLDER NAME

GIT STATUS –S

GIT DIFF ->STAGED AND NON STAGED

GIT COMMIT –V

GIT FETCH [remote name]->FETCH FILES FROM REMOTE WHICH ARE NOT ADDED WEN COLED

GIT PULL -> FETCH FILES AND MERGES FROM REMOTE WHICH ARE NOT ADDED WEN COLED

GIT PUSH ->push to the server

 git push [remote-name] [branch-name].

$ git log --oneline --decorate --graph --all

\* c2b9e (HEAD, master) made other changes

| \* 87ab2 (testing) made a change

|/

git branch testing

git checkout testing

I scenario

Git clone, add,commit,push to remote

2 scenario

Branch, do changes

Git branch

Checkout

Pull

commit

Push to remote branch

$ git push --set-upstream origin origin/uat

3 .scenario

Local branch of master , no remote branch

Do changes , commit, merge

Push to master

**BOOTSTRAP**

**12 column layout**

**.col-xs-4**

**.col-md-5**

**SASS**

1.**VARIABLES**

$variablename : value

$cool-red:#123455

$box-shadow-bottom-only: 0 2px 1px 0 rgba(0, 0, 0, 0.2);

H1.title

{

Color : $cool-red

Box-shadow : $box-shadow-bottom-only

}

**2.mixin ->constructor**

@mixin textfield($ht,$wt)

{

Height:$ht;

Width:$wt

}

.name

{

@include textfield(10,20)

}

.email

{

@include **textfield(20,30)**

**}**

**3.extends**

.dialog

{ Height:10;

Width:20

}

.dialog-true

{

@extends .dialog

Color:green

}

**4.Nesting**