

Building Hypermedia APIs in Javascript

from homemade solution to using Fortune.js

You?



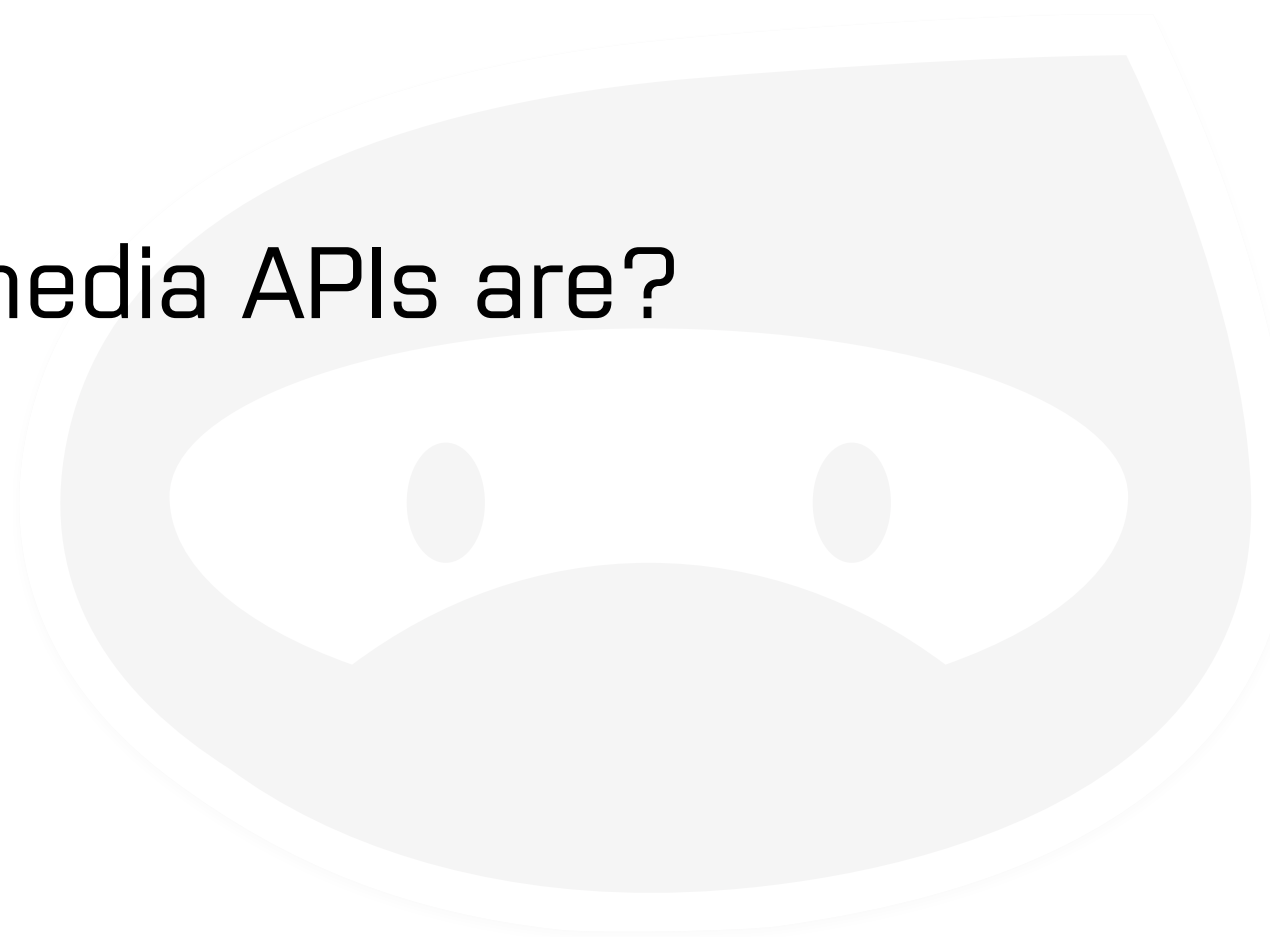
You?

- Who uses APIs?



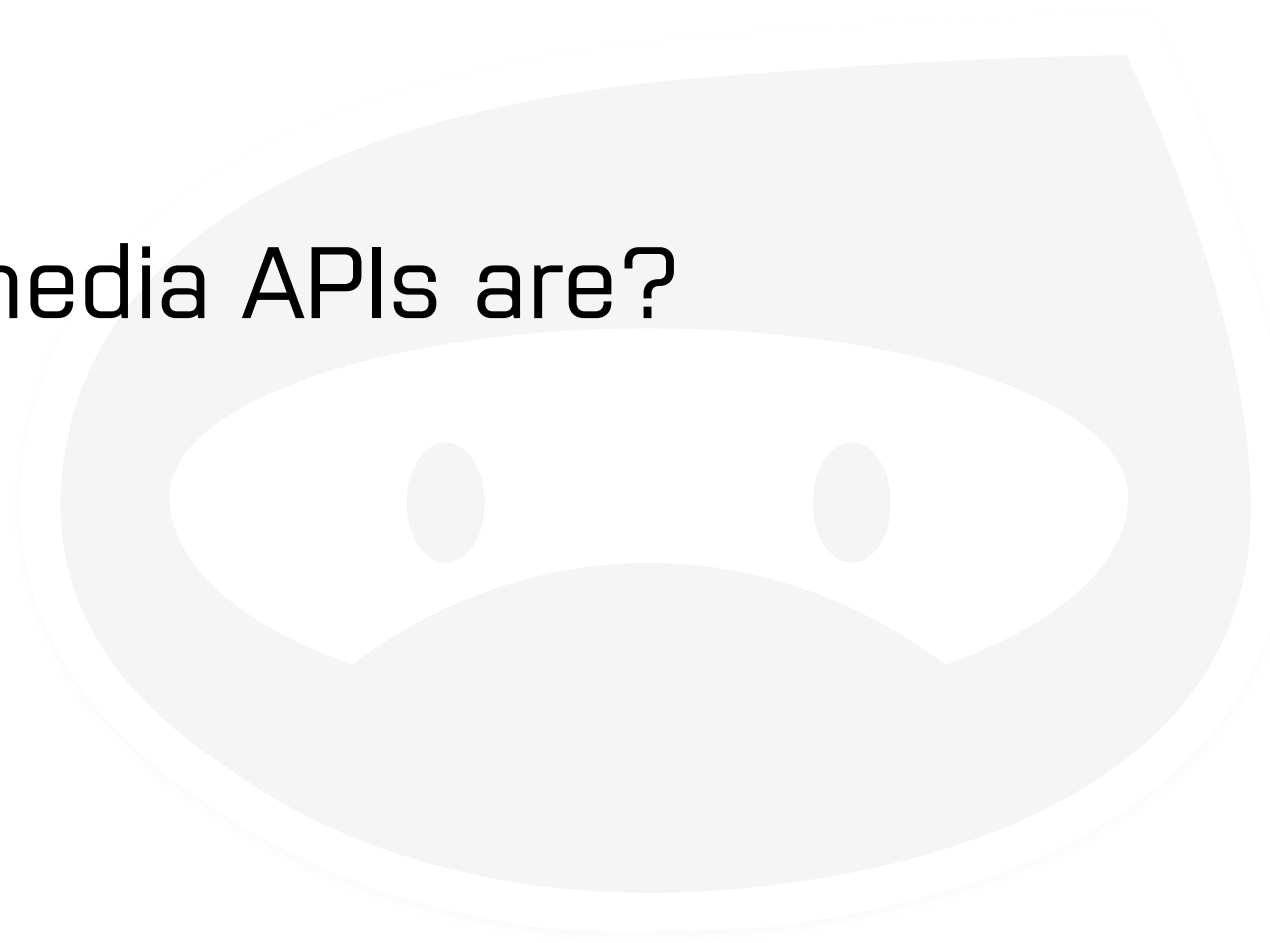
You?

- Who uses APIs?
- Who knows what Hypermedia APIs are?



You?

- Who uses APIs?
- Who knows what Hypermedia APIs are?
- Who had ever build one?



Me



Me

- 3scale evangelist



Me

- 3scale evangelist
- API freak



Me

- 3scale evangelist
- API freak
- Hacker in Residence



Me

- 3scale evangelist
- API freak
- Hacker in Residence
- Node.js, Meteor



Hypermedia 101

you are already using Hypermedia APIs

PayPal



huddle

Core values



Core values

- Discoverable API



Core values

- Discoverable API
- Machine readable API



Core values

- Discoverable API
- Machine readable API
- Link-based



Example: Github

```
GET api.github.com
{
  current_user_url: "https://api.github.com/user",
  authorizations_url: "https://api.github.com/authorizations",
  code_search_url: "https://api.github.com/search/code?q={query}
{&page,per_page,sort,order}",
  emails_url: "https://api.github.com/user/emails",
  emojis_url: "https://api.github.com/emojis",
  events_url: "https://api.github.com/events",
  feeds_url: "https://api.github.com/feeds",ed{/owner}/{/repo}",
...
  team_url: "https://api.github.com/teams",
  user_url: "https://api.github.com/users/{user}",
  user_organizations_url: "https://api.github.com/user/orgs",
  user_repositories_url: "https://api.github.com/users/{user}/
repos{?type,page,per_page,sort}",
  user_search_url: "https://api.github.com/search/users?q={query}
{&page,per_page,sort,order}"
}
```

Example: building

REST approach

/rooms/r123

/doors/d123



Example: building

Adding relations

/rooms/r123

- /rooms/r123/doors/d123

/doors/d123

- /doors/d123/rooms/r123



Example: building

Buildings have hallways

/doors/1/rooms/1

/doors/1/hallways/1

/rooms/1/doors/1

/rooms/1/hallways/1

/hallways/1/doors/1

/hallways/1/rooms/1



Me...



Me...



Let's put Hypermedia in it

Linking ressources

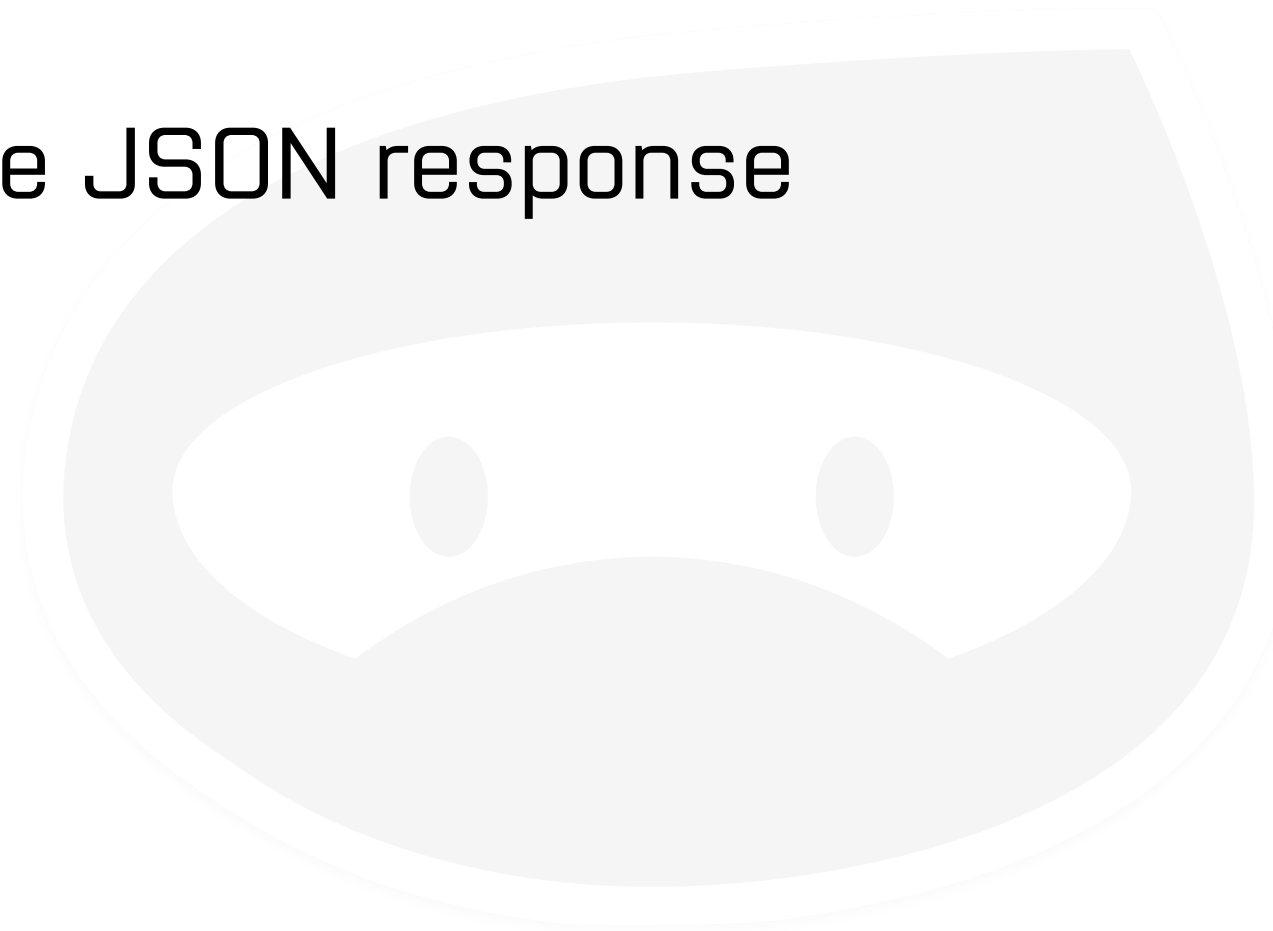
- /rooms/1
 - **rel:** door, **href:** /what/ever/door
 - **rel:** door, **href:** /another/door
- /door
 - **rel:** entry-to, **href:** /rooms/1
 - **rel:** entry-to, **href:** /hallways/1
- /hallways/1
 - **rel:** floor, **href:** /floors/1



HATEOAS

Hypermedia as the Engine of Application State

- Basic implementation with Express
- add "link" properties to the JSON response

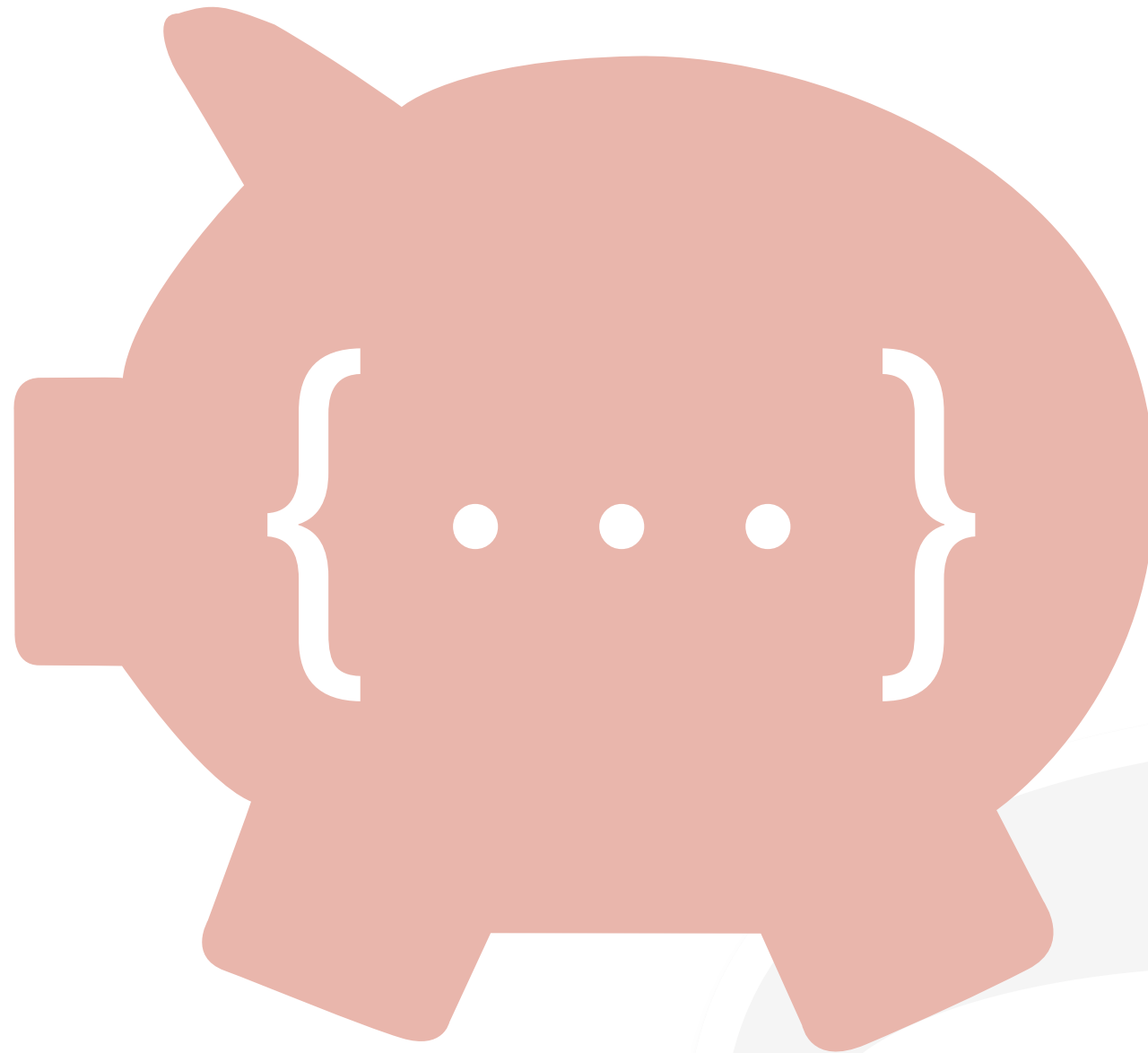


HATEOAS

```
GET mybuilding.com/rooms/1
{
  "data": {
    "name": "Kitchen",
    "id": "1",
    "links" : [
      {
        "rel": "door",
        "href": "https://mybuilding.com/rooms/12345678"
      }, {
        "rel": "door",
        "href": "https://theirbuilding.com/rooms/
12345678"
      } ]
    }
  }
}
```

There is a framework for that





Fortune.js

Fortune.js

- simple to setup

```
var fortune = require('fortune')
, app = fortune({
  db: 'petstore'
})
.resource('person', {
  name: String,
  age: Number,
  pets: ['pet'] // "has many" relationship to pets
})
.resource('pet', {
  name: String,
  age: Number,
  owner: 'person' // "belongs to" relationship to a person
})
.listen(1337);
```

Fortune.js

HTTP	Person	Pet	Notes
GET	/people	/pets	Get a collection of resources,
POST	/people	/pets	Create a resource
GET	/people/:id	/pets/:id	Get a specific resource, or
PUT	/people/:id	/pets/:id	Create or update a resource
PATCH	/people/:id	/pets/:id	Patch a resource (see
DELETE	/people/:id	/pets/:id	Delete a resource
GET	/people/:id/pets	/pets/:id/owner	Get a related resource (one

Fortune.js

- JSON API spec

```
{
  "people": [
    {
      "id": "tO69XnLz7amaYVRq",
      "name": "Wall-E",
      "age": 1000,
      "links": {
        "pets": [
          "dPuXqUJxRIn9cOdH"
        ]
      }
    }
  ],
  "links": {
    "people.pets": {
      "href": "/pets/{people.pets}",
      "type": "pets"
    }
  }
}
```

Fortune.js

- JSON API spec

```
/people/tO69XnLz7amaYVRq
{
  "people": [
    {
      "id": "tO69XnLz7amaYVRq",
      "name": "Wall-E",
      "age": 1000,
      "links": {
        "pets": [
          "dPuXqUJxRIn9cOdH"
        ]
      }
    }
  ],
  "links": {
    "people.pets": {
      "href": "/pets/{people.pets}",
      "type": "pets"
    }
  }
}
```



Fortune.js

- JSON API spec

```
/people/tO69XnLz7amaYVRq
{
  "people": [
    {
      "id": "tO69XnLz7amaYVRq",
      "name": "Wall-E",
      "age": 1000,
      "links": {
        "pets": [
          "dPuXqUJxRIn9cOdH"
        ]
      }
    }
  ],
  "links": {
    "people.pets": {
      "href": "/pets/{people.pets}",
      "type": "pets"
    }
  }
}
```

```
/pets/dPuXqUJxRIn9cOdH
{
  "pets": [
    {
      "id": "dPuXqUJxRIn9cOdH",
      "name": "Cucaracha",
      "age": 1000,
      "links": {
        "owner": "tO69XnLz7amaYVRq"
      }
    }
  ],
  "links": {
    "pets.owner": {
      "href": "/people/{pets.owner}",
      "type": "people"
    }
  }
}
```


Fortune.js

- ☒ adapters for NeDB, MongoDB, MySQL, Postgres, SQLite
- ☒ handles all the routing
- ☒ handles database interactions
- ☒ hooks for specific logic before/after interacting with resources
- ☐ authentication not implemented (build your own)

API-based game: APIbunny

APIbunny

```
{  
  north: { ref: 'cell', inverse: 'south' },  
  east: { ref: 'cell', inverse: 'west' },  
  south: { ref: 'cell', inverse: 'north' },  
  west: { ref: 'cell', inverse: 'east' }  
}
```



APIbunny

- Inverse relation does not work in case of maze

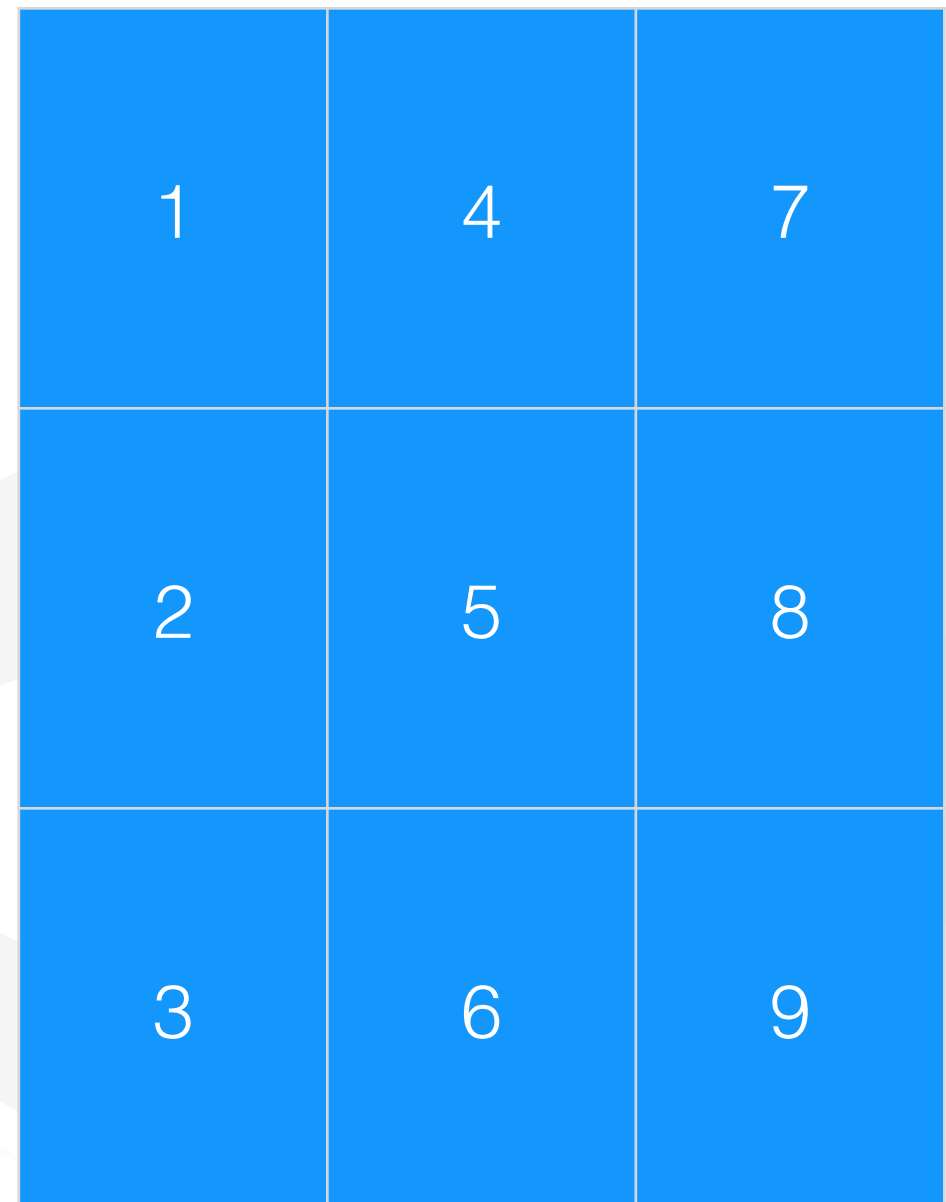
```
{  
  north: { ref: 'cell', inverse: 'south' },  
  east: { ref: 'cell', inverse: 'west' },  
  south: { ref: 'cell', inverse: 'north' },  
  west: { ref: 'cell', inverse: 'east' }  
}
```



APIbunny

- Inverse relation does not work in case of maze

```
{
  north: { ref: 'cell', inverse: 'south' },
  east: { ref: 'cell', inverse: 'west' },
  south: { ref: 'cell', inverse: 'north' },
  west: { ref: 'cell', inverse: 'east' }
}
```



1	4	7
2	5	8
3	6	9

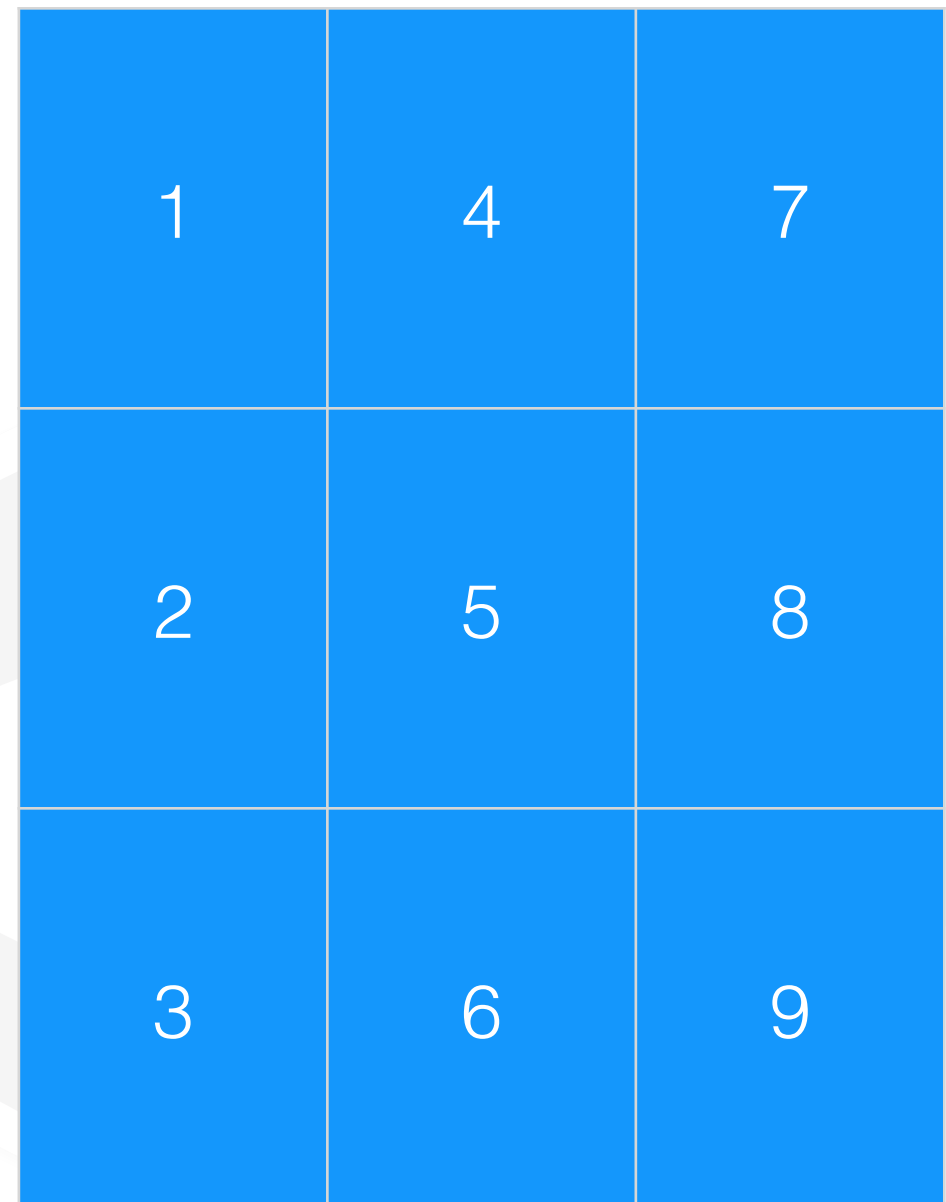
APIbunny

- Inverse relation does not work in case of maze

```
{  
  north: { ref: 'cell', inverse: 'south' },  
  east: { ref: 'cell', inverse: 'west' },  
  south: { ref: 'cell', inverse: 'north' },  
  west: { ref: 'cell', inverse: 'east' }  
}
```

expected on #4

```
{  
  west:1  
  east:7  
  south:5  
}
```



1	4	7
2	5	8
3	6	9

APIbunny

- Inverse relation does not work in case of maze

```
{
  north: { ref: 'cell', inverse: 'south' },
  east: { ref: 'cell', inverse: 'west' },
  south: { ref: 'cell', inverse: 'north' },
  west: { ref: 'cell', inverse: 'east' }
}
```

expected on #4

```
{
  west:1
  east:7
  south:5
}
```

east of #1 is #4 -> west of #4 is #1

but east of #7 is #4 -> west of #4 is #7

1	4	7
2	5	8
3	6	9

APIbunny

- POST requests format is not usual

```
{ "people": [ { "age": 1000, "name": "Wall-E" } ] }
```



APIbunny

- thousands players, 44 winners, ~10 code shared
- Open <https://github.com/picsoung/apibunny/>
- V2 is coming ;)



Ressources

- <http://apicodex.3scale.net/>
- <http://amundsen.com/hypermedia/>
- <http://www.designinghypermediaapis.com/>
- https://www.youtube.com/watch?v=_UG7u7ARTfM
[APIStrat SF]
- APIdays and APIstrat conferences
- API-craft meetup + google group

Questions?

 picsoung

 picsoung