JavaScript "this"

this

```
var hero = {
  getInventory: function () {
    return this;
  }
};

assertEquals(hero.getInventory(), hero);
```

this

- points to the object
  the method is "called on"

this

```
var hank = {
  name: "Hank",
  speak: function () {
    console.log("Hi my name is " + this.name);
  }
}

hank.speak(); // Hi my name is Hank


var speak = hank.speak;

speak(); // Throws: Cannot read name of undefine
```

this

```
//...

var becca = {
  name: "Becca"
};


becca.speak = hank.speak;

becca.speak(); // Hi my name is Becca
```

this

- Dynamic
- Value decided at call time
- Constant within an *execution context*

Function.prototype.bind

```
var hank = {
  name: "Hank",
  speak: function () {
    console.log("Hi my name is " + this.name);
  }
}

hank.speak(); // Hi my name is Hank


var speak = hank.speak;

speak(); // Throws: Cannot read name of undefine
```

Function.prototype.bind

```
var hank = {
  name: "Hank",
  speak: function () {
    console.log("Hi my name is " + this.name);
  }
}
var speak = hank.speak.bind(hank);



speak(); // Hi my name is Hank
```

Function.prototype.bind

```
function sum() {
  // sums every argument passed..
}

var partiallyAppliedSum = sum.bind(null, 1, 2, 3);



partiallyAppliedSum(); // 6
```

Function.prototype.bind

```
function sum(a, b) {
  return a + b;
}

var add5 = sum.bind(null, 5);

// function add5(b) {
//   return 5 + b;
// }

add5(10); // 15
```

Partial application!

Function.prototype.bind

- Returns a new function where this is fixed
- Once bound, always bound
- Can also be used to partially apply arguments
- Without actually executing a function

This Exercises

Canvas: exercises-this.zip