

Programmation web - Client riche

this

```
class User {  
  constructor(name) {  
    this.name = name  
  }  
  
  greet() {  
    console.log("Hello, " + this.name)  
  }  
}
```

```
const user = new User("Marin")
```

```
user.greet()  
// => "Hello, Marin"
```

```
class User {  
  constructor(name) {  
    this.name = name  
  }  
  
  greet() {  
    console.log("Hello, " + this.name)  
  }  
}
```

```
const user = new User("Marin")
```

```
user.greet()  
// => "Hello, Marin"
```

```
const greet = user.greet  
console.log(greet)  
// => function greet()
```

```
class User {  
  constructor(name) {  
    this.name = name  
  }  
  
  greet() {  
    console.log("Hello, " + this.name)  
  }  
}
```

```
const user = new User("Marin")
```

```
user.greet()  
// => "Hello, Marin"
```

```
const greet = user.greet  
console.log(greet)  
// => function greet()
```

```
greet()  
// => TypeError: this is undefined
```

```
class MyElement {
  constructor(type) {
    this.type = type

    this.eventHandlers = {}
  }

  addEventListener(eventType, handler) {
    this.eventHandlers[eventType] = [
      ...this.eventHandlers[eventType],
      handler
    ]
  }

  dispatchEvent(eventType) {
    const handlers = this.eventHandlers[eventType] || []

    for (const handler of handlers) {
      handler()
    }
  }
}
```

```
class MyComponent {  
  constructor() {  
    this.root = new MyElement("div")  
  
    this.onInit()  
  }  
  
  onInit() {  
    this.root.addEventListener("click", this.onClick)  
  }  
  
  onClick() {  
    console.log(this)  
  }  
}
```

```

class MyComponent {
  //...

  onInit() {
    this.root.addEventListener("click" this.onClick)
  }

  // ...
}

```

```

class MyElement {
  // ...

```

```

  addEventListener(eventType, handler) {
    this.eventHandlers[eventType] = [
      ...this.eventHandlers[eventType],
      handler
    ]
  }

  dispatchEvent(eventType) {
    const handlers = this.eventHandlers[eventType] || []

    for (const handler of handlers) {
      handler()
    }
  }
}

```

const greet = user.greet
=> « perte » du contexte `this`

Appel de la fonction sans
contexte `this`

Attacher un contexte à une fonction


```
function logName() {  
  console.log(this.name)  
}
```

```
const user = {  
  name: "Marin"  
}
```

```
const boundLogName = logName.bind(user)  
boundLogName() // => "Marin"
```

```
logName.call(user) // => "Marin"  
logName.apply(user) // => "Marin"
```

```

class MyComponent {
  constructor() {
    this.root = new MyElement("div")
    this.onClick = this.onClick.bind(this)
  }

  onInit() {
    this.root.addEventListener("click", this.onClick)
  }

  // ...
}

class MyElement {
  // ...

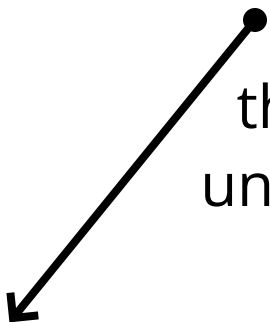
  addEventListener(eventType, handler) {
    this.eventHandlers[eventType] = [
      ...this.eventHandlers[eventType],
      handler
    ]
  }

  dispatchEvent(eventType) {
    const handlers = this.eventHandlers[eventType] || []

    for (const handler of handlers) {
      handler()
    }
  }
}

```

this.onClick est attaché à
un contexte particulier, son
this est fixé



Appel de la fonction, qui connaît
son this

Des questions ?

- L'opérateur this
- Understanding the "this" keyword, call, apply, and bind in JavaScript