JavaScript

STUDIA PODYPLOMOWE
POLITECHNIKA BIAŁOSTOCKA

meet.js
białystok

meet.js Białystok #57

#javascript #front-end #meetup #białystok
#networking #prezentacje #piwo

28.05.24 // 18:00 // Klub Gwint
ul. Zwierzyniecka 10

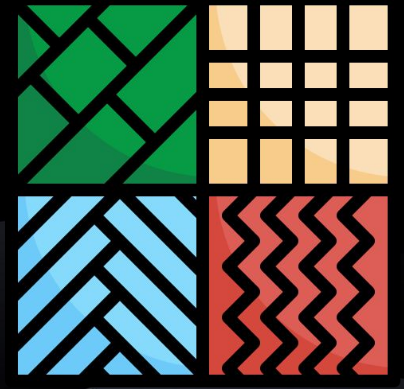
spyrosoft

partner wydarzenia

#4

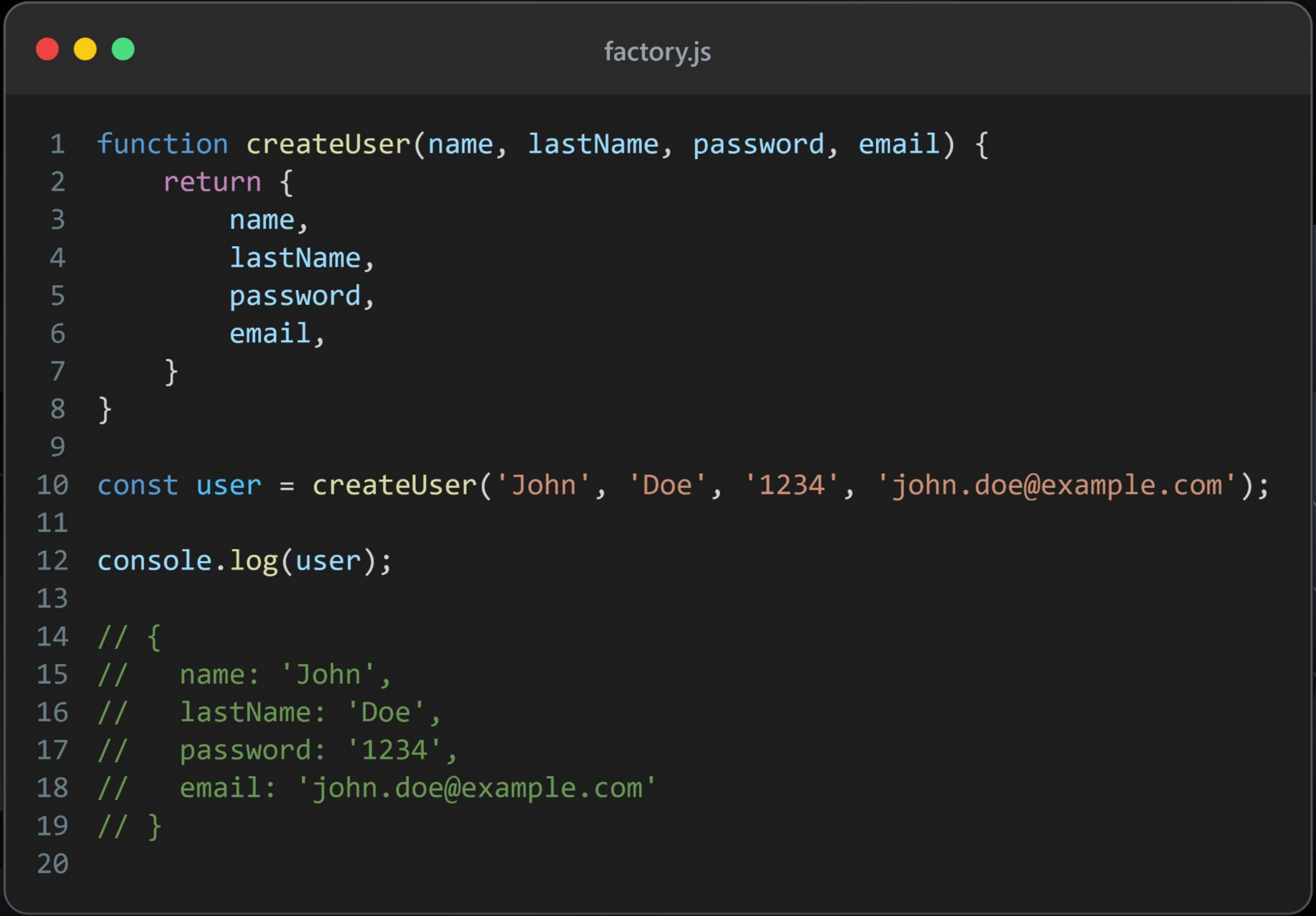
Design Patterns 2

with JavaScript



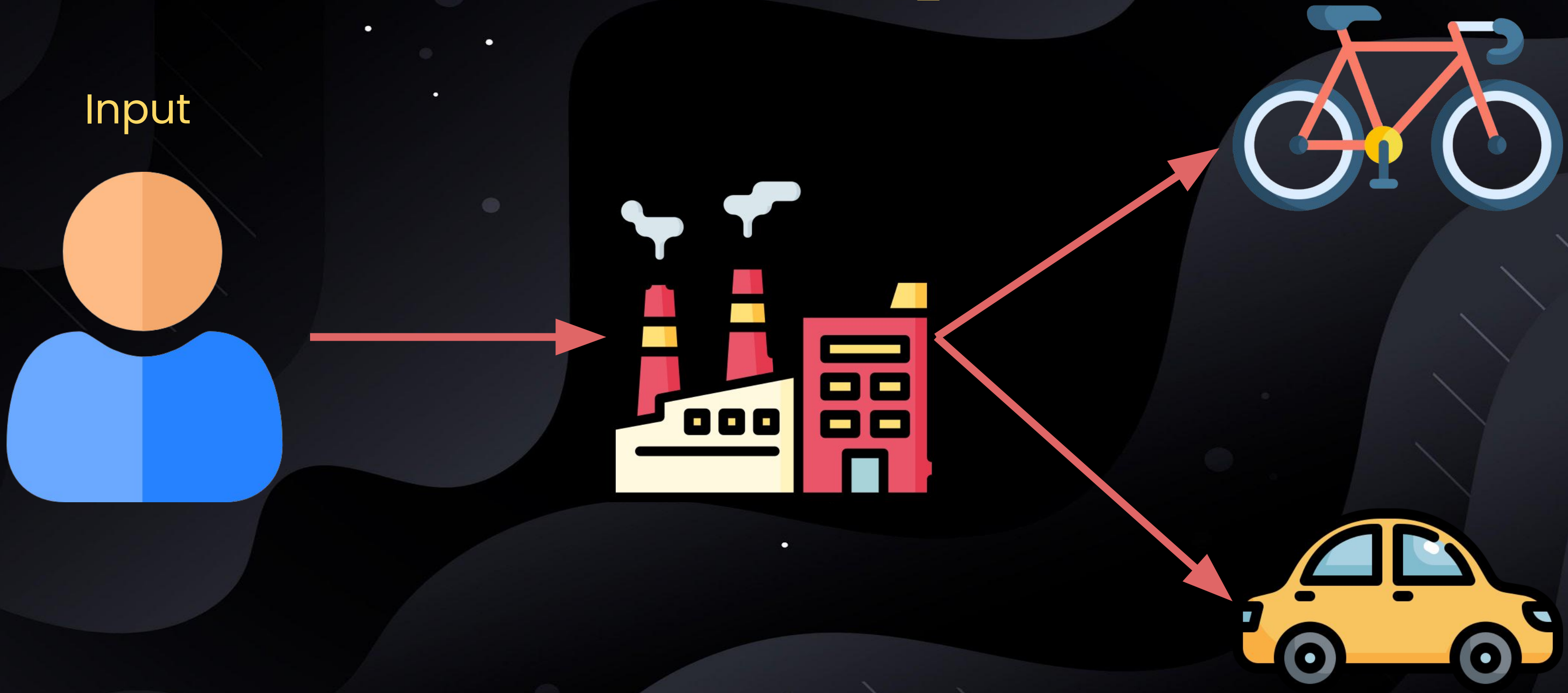
Factory

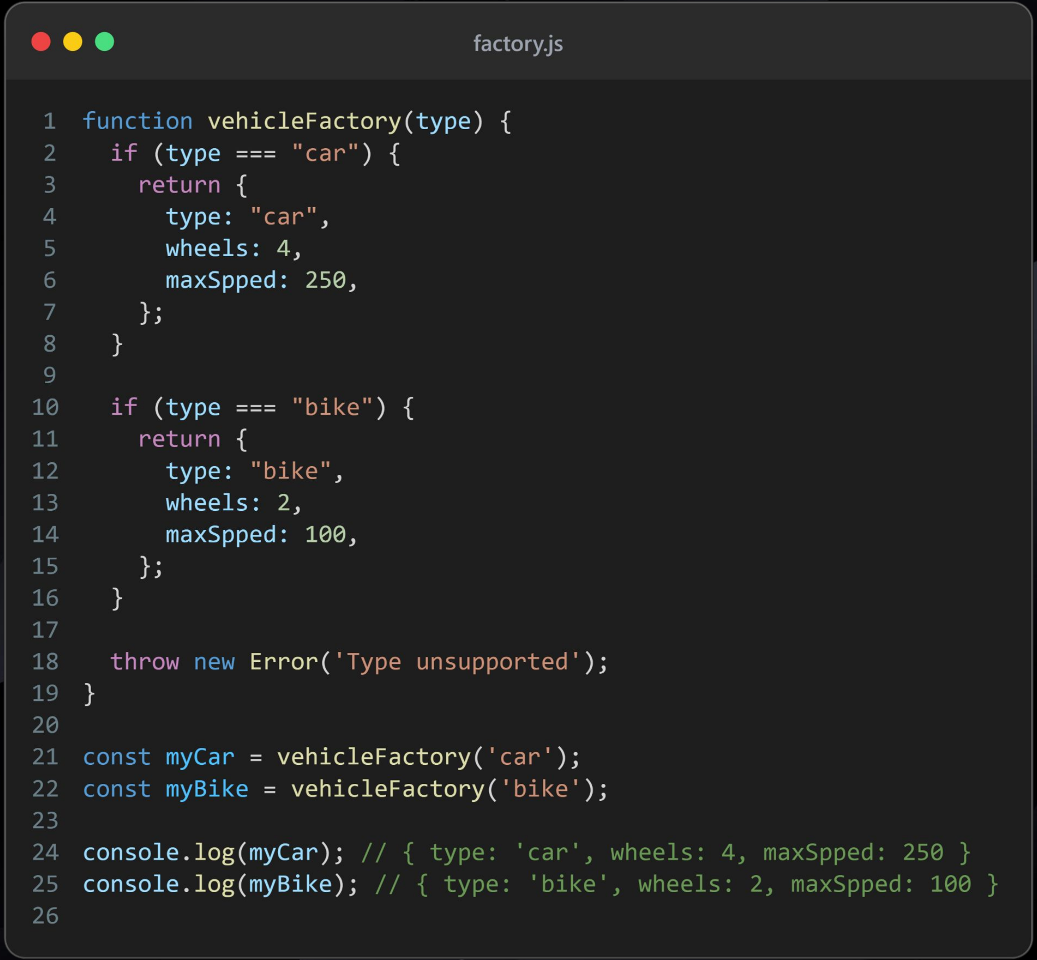
- At it’s core - a function that returns an object
- Base on input it returns different objects


factory.js

```javascript
function createUser(name, lastName, password, email) {
    return {
        name,
        lastName,
        password,
        email,
    }
}

const user = createUser('John', 'Doe', '1234', 'john.doe@example.com');

console.log(user);

// {
//   name: 'John',
//   lastName: 'Doe',
//   password: '1234',
//   email: 'john.doe@example.com'
// }

```

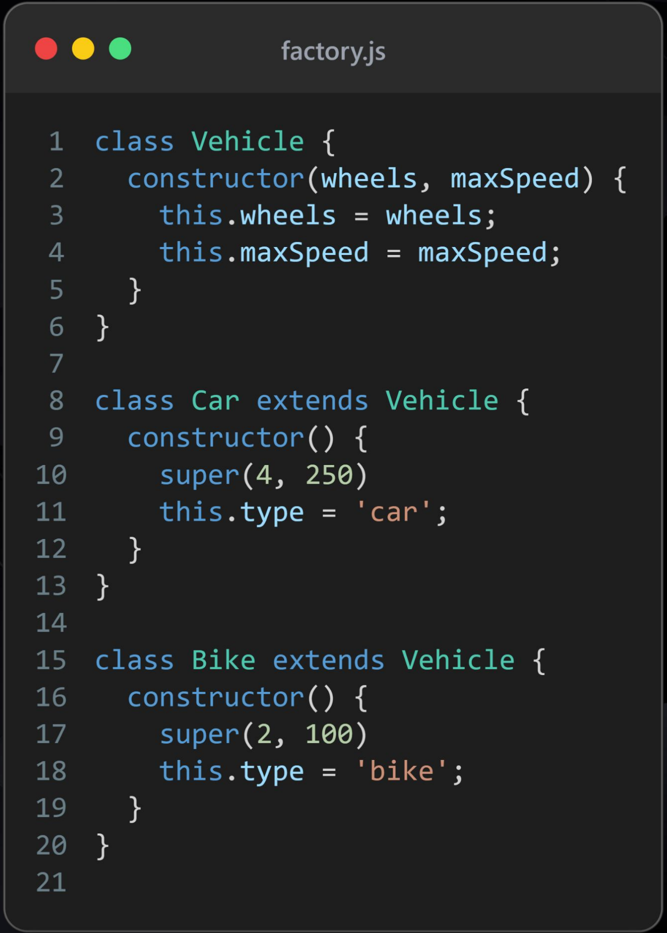
Factory



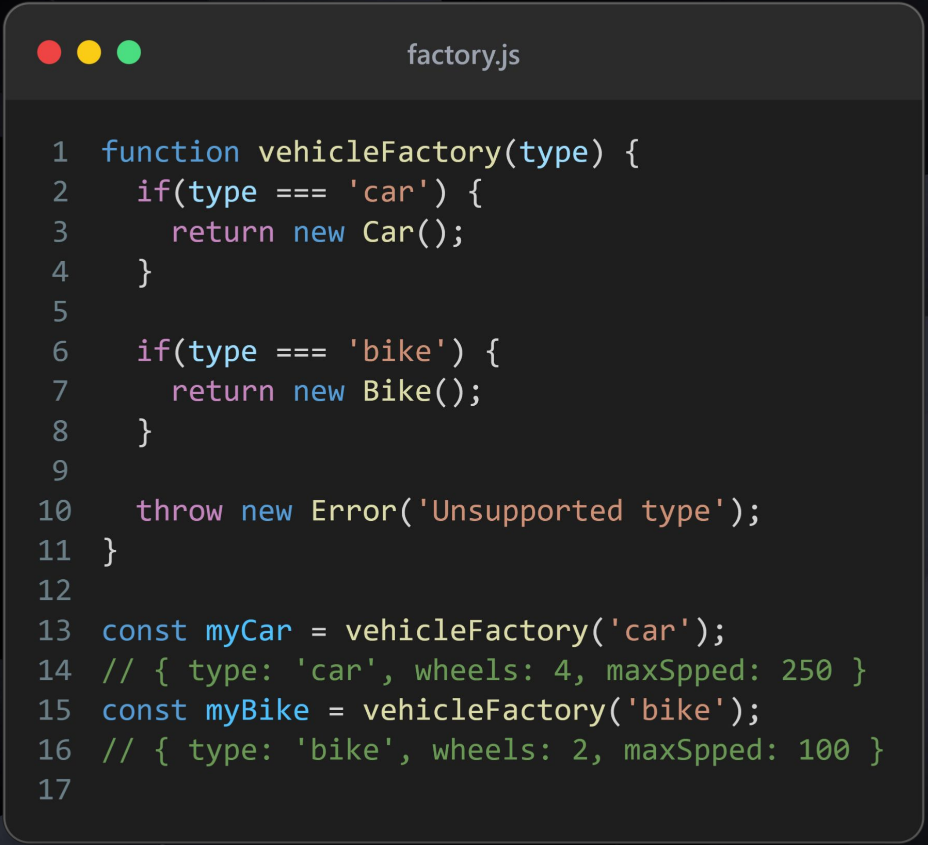


factory.js

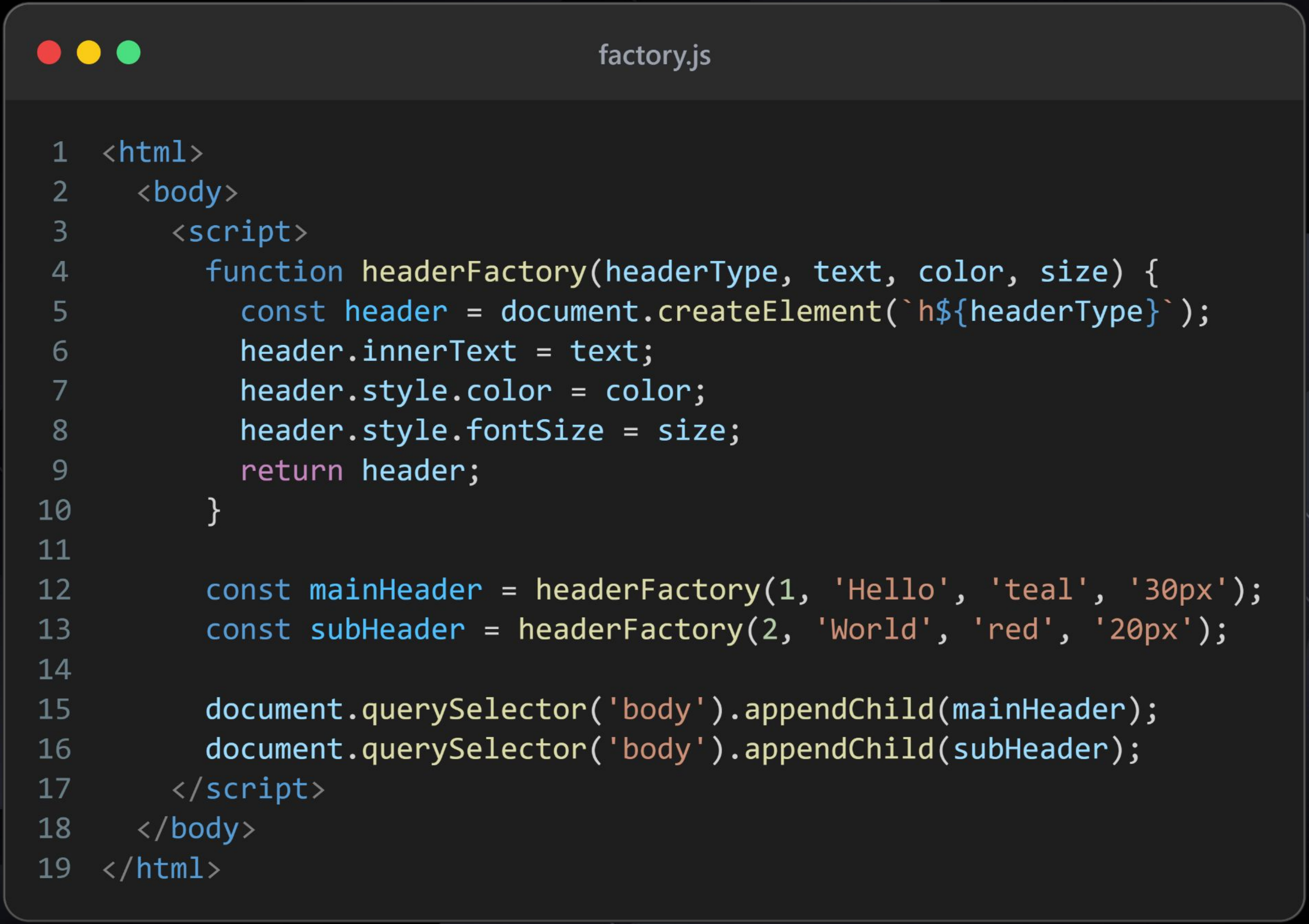
```js
function vehicleFactory(type) {
  if (type === "car") {
    return {
      type: "car",
      wheels: 4,
      maxSpped: 250,
    };
  }

  if (type === "bike") {
    return {
      type: "bike",
      wheels: 2,
      maxSpped: 100,
    };
  }

  throw new Error('Type unsupported');
}

const myCar = vehicleFactory('car');
const myBike = vehicleFactory('bike');

console.log(myCar); // { type: 'car', wheels: 4, maxSpped: 250 }
console.log(myBike); // { type: 'bike', wheels: 2, maxSpped: 100 }
```



```javascript
class Vehicle {
  constructor(wheels, maxSpeed) {
    this.wheels = wheels;
    this.maxSpeed = maxSpeed;
  }
}

class Car extends Vehicle {
  constructor() {
    super(4, 250)
    this.type = 'car';
  }
}

class Bike extends Vehicle {
  constructor() {
    super(2, 100)
    this.type = 'bike';
  }
}

```



```javascript
function vehicleFactory(type) {
  if(type === 'car') {
    return new Car();
  }

  if(type === 'bike') {
    return new Bike();
  }

  throw new Error('Unsupported type');
}

const myCar = vehicleFactory('car');
// { type: 'car', wheels: 4, maxSpped: 250 }
const myBike = vehicleFactory('bike');
// { type: 'bike', wheels: 2, maxSpped: 100 }

```

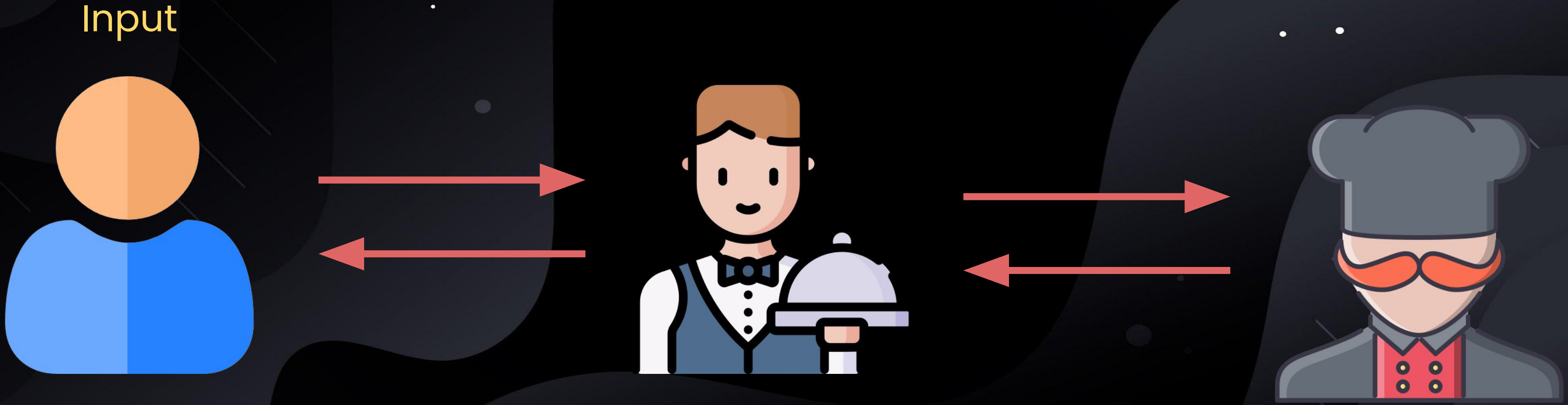


```
<html>
  <body>
    <script>
      function headerFactory(headerType, text, color, size) {
        const header = document.createElement(`h${headerType}`);
        header.innerText = text;
        header.style.color = color;
        header.style.fontSize = size;
        return header;
      }

      const mainHeader = headerFactory(1, 'Hello', 'teal', '30px');
      const subHeader = headerFactory(2, 'World', 'red', '20px');

      document.querySelector('body').appendChild(mainHeader);
      document.querySelector('body').appendChild(subHeader);
    </script>
  </body>
</html>
```

Proxy

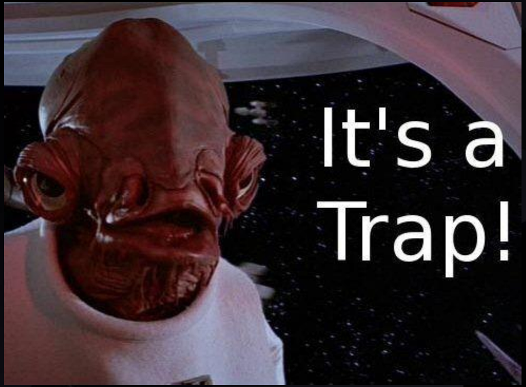
- An object that is a middleman when interacting with the original Object
- Used to alter or augment the original object’s behavior
- Usually used to augment get or set

Proxy



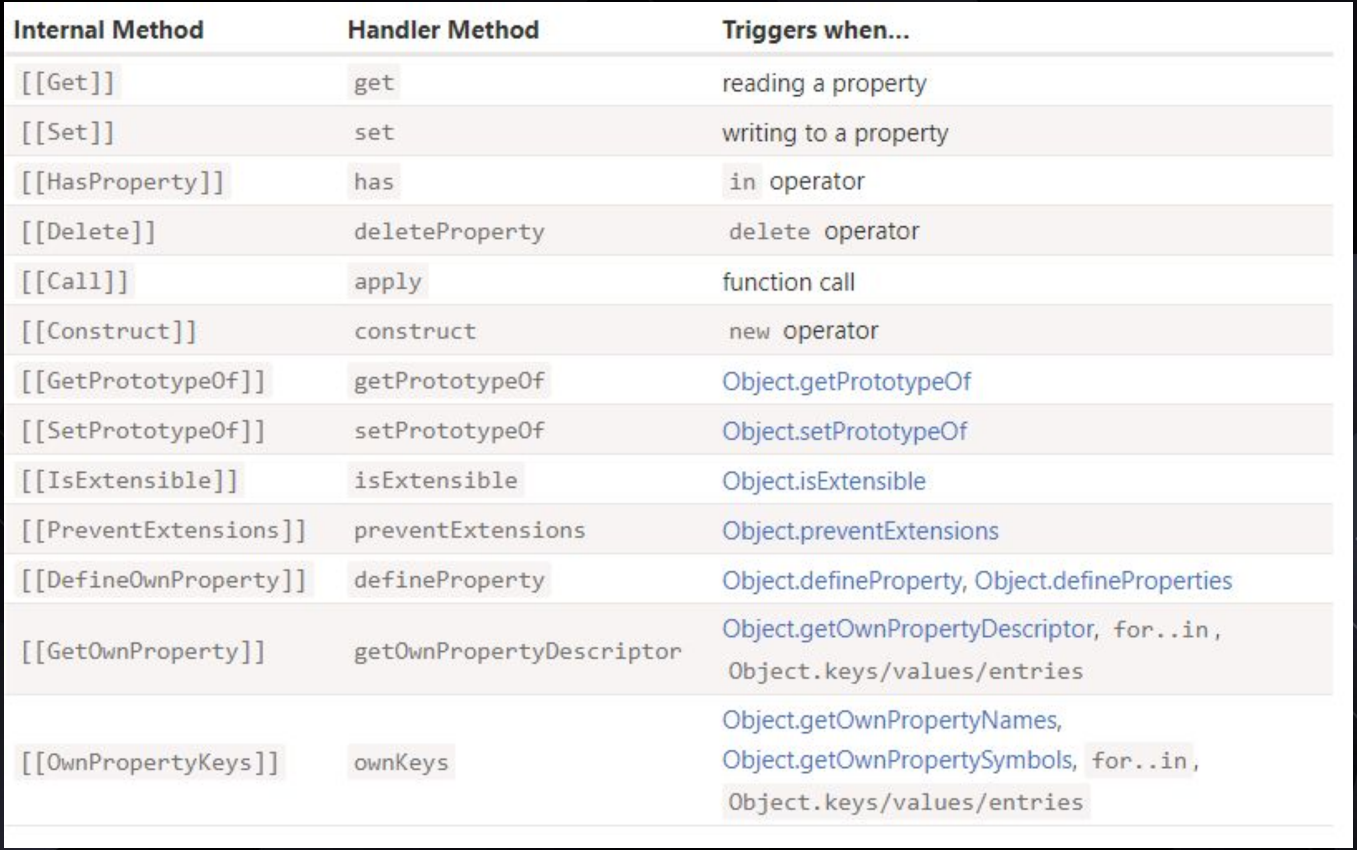
new Proxy

- Accept two parameters
- First is the original (target) object that will be wrapped in proxy
- Second is proxy configuration object. This object can have “traps”, methods that intercept operations



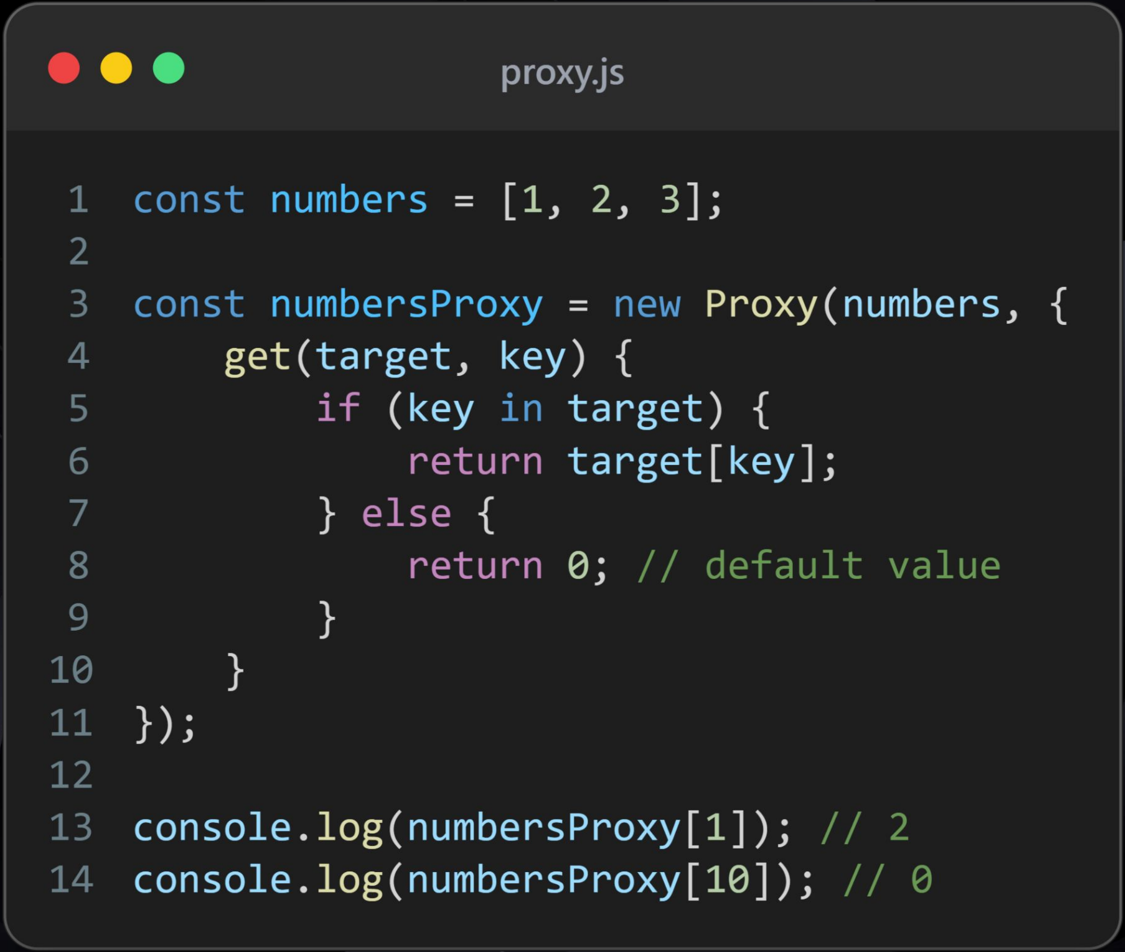
Proxy traps

- Proxy traps intercept invocation of object internal methods
- Internal methods are only used in the specification, we can’t call them directly by name

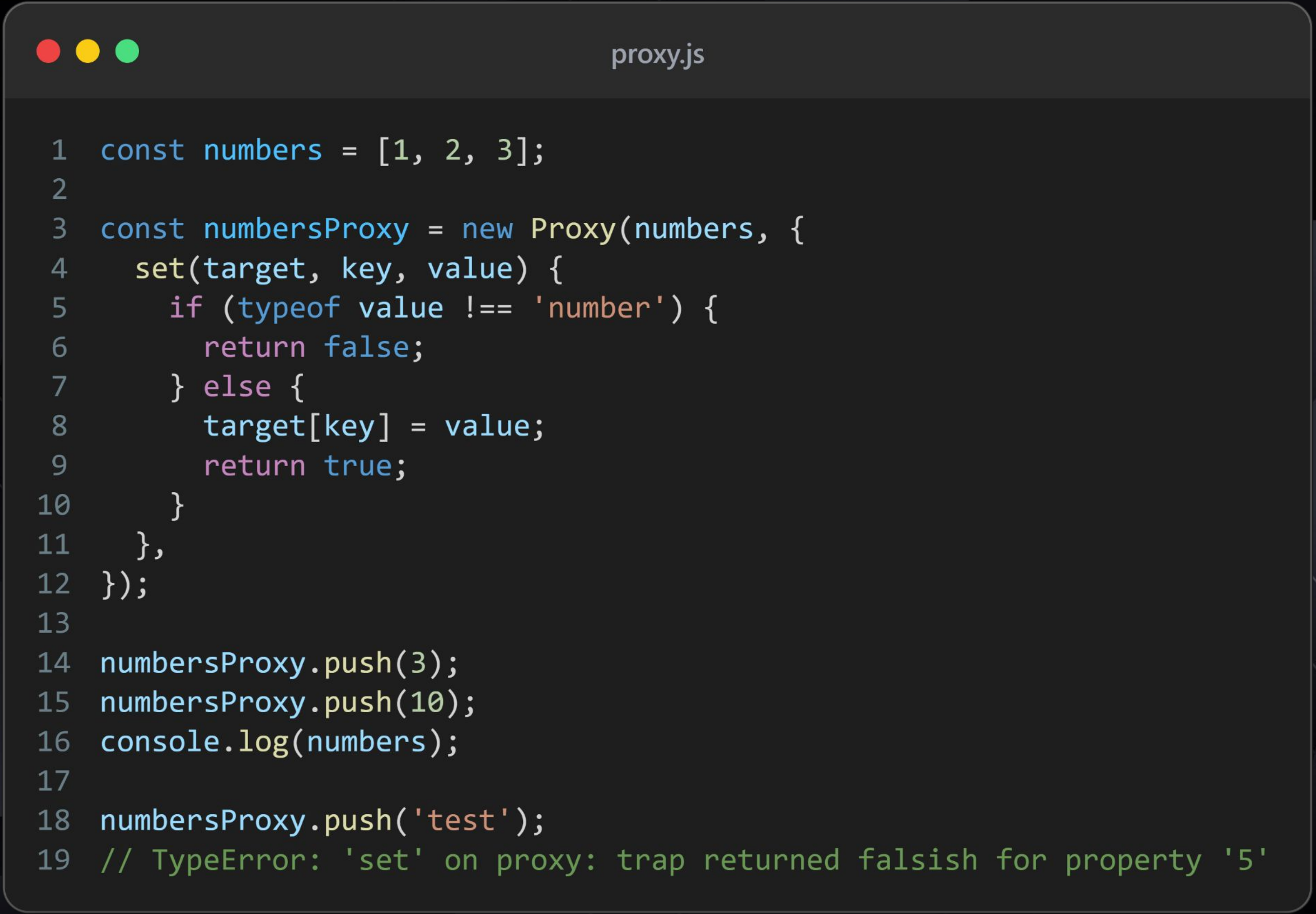


Internal Method	Handler Method	Triggers when...
`[[Get]]`	`get`	reading a property
`[[Set]]`	`set`	writing to a property
`[[HasProperty]]`	`has`	`in` operator
`[[Delete]]`	`deleteProperty`	`delete` operator
`[[Call]]`	`apply`	function call
`[[Construct]]`	`construct`	`new` operator
`[[GetPrototypeOf]]`	`getPrototypeOf`	Object.getPrototypeOf
`[[SetPrototypeOf]]`	`setPrototypeOf`	Object.setPrototypeOf
`[[IsExtensible]]`	`isExtensible`	Object.isExtensible
`[[PreventExtensions]]`	`preventExtensions`	Object.preventExtensions
`[[DefineOwnProperty]]`	`defineProperty`	Object.defineProperty, Object.defineProperties
`[[GetOwnProperty]]`	`getOwnPropertyDescriptor`	Object.getOwnPropertyDescriptor, `for..in`, `Object.keys/values/entries`
`[[OwnPropertyKeys]]`	`ownKeys`	Object.getOwnPropertyNames, Object.getOwnPropertySymbols, `for..in`, `Object.keys/values/entries`

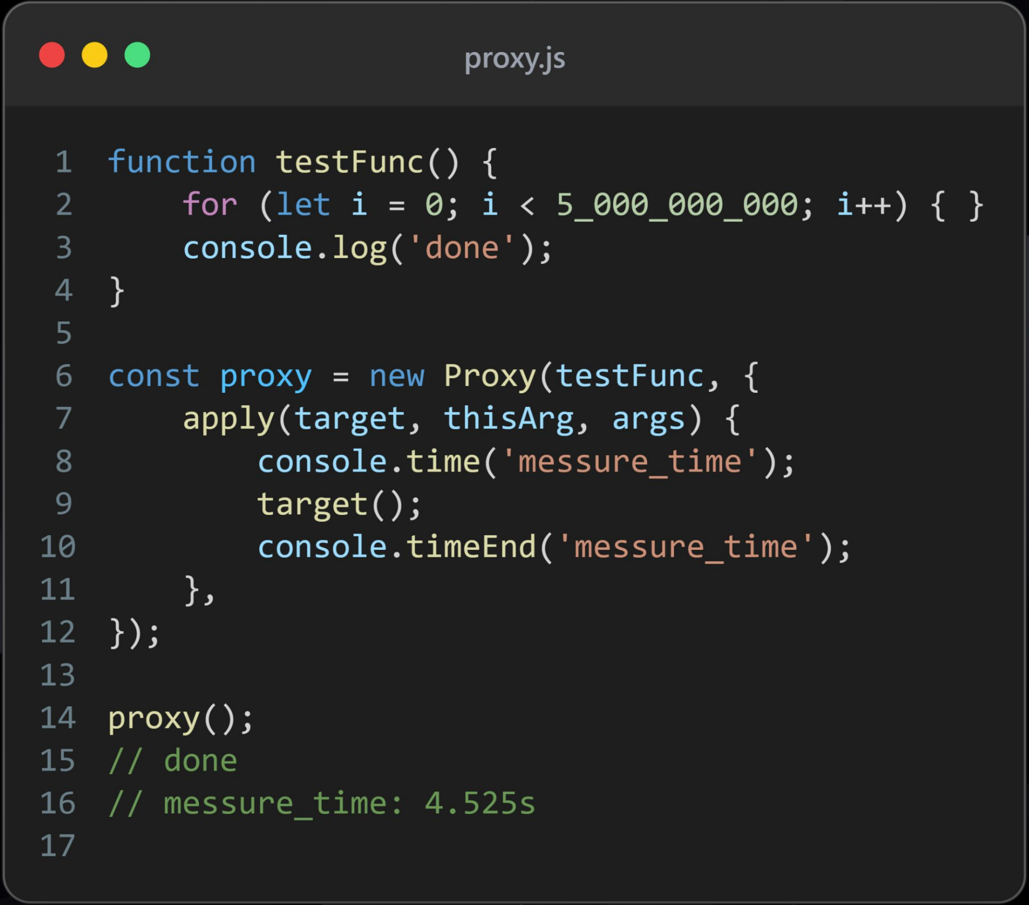
javascript.info - Proxy


proxy.js

```javascript
const numbers = [1, 2, 3];

const numbersProxy = new Proxy(numbers, {
    get(target, key) {
        if (key in target) {
            return target[key];
        } else {
            return 0; // default value
        }
    }
});

console.log(numbersProxy[1]); // 2
console.log(numbersProxy[10]); // 0
```

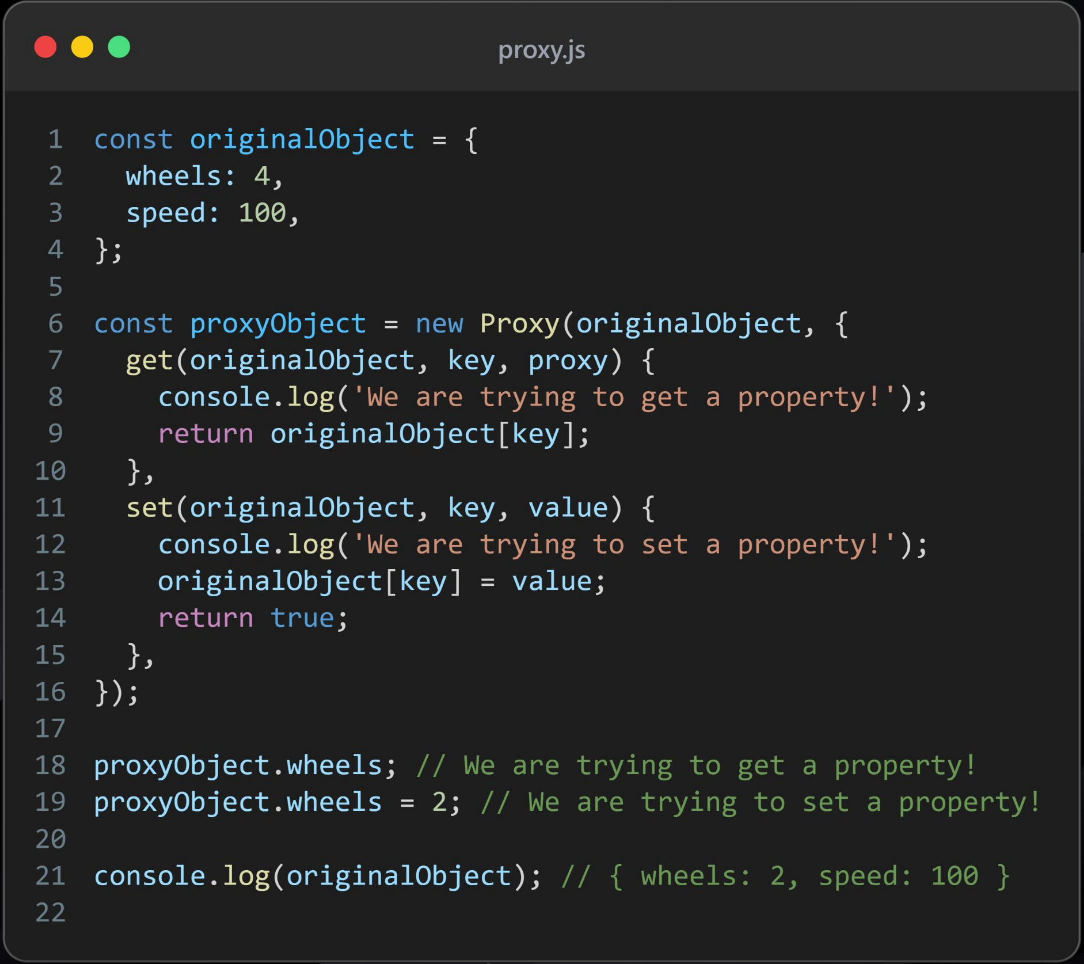

proxy.js

```javascript
const numbers = [1, 2, 3];

const numbersProxy = new Proxy(numbers, {
  set(target, key, value) {
    if (typeof value !== 'number') {
      return false;
    } else {
      target[key] = value;
      return true;
    }
  },
});

numbersProxy.push(3);
numbersProxy.push(10);
console.log(numbers);

numbersProxy.push('test');
// TypeError: 'set' on proxy: trap returned falsish for property '5'
```



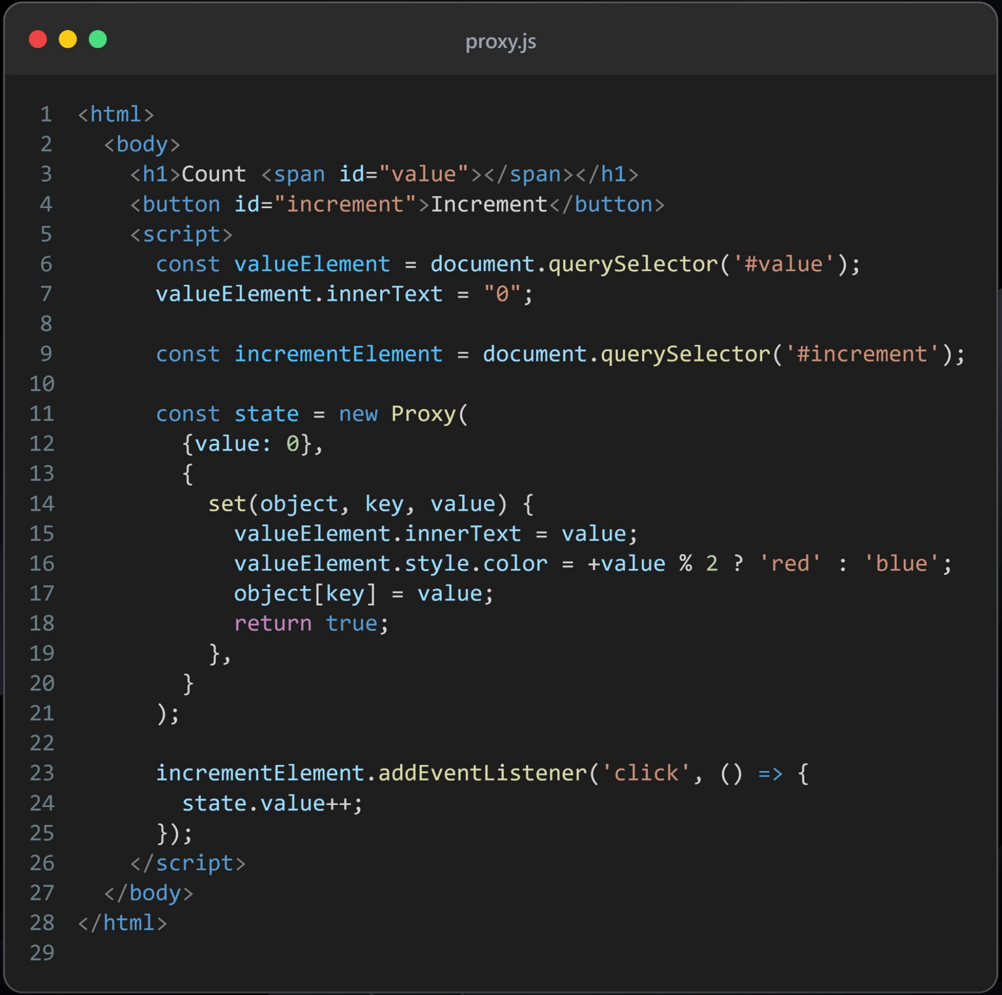
proxy.js

```
function testFunc() {
    for (let i = 0; i < 5_000_000_000; i++) { }
    console.log('done');
}

const proxy = new Proxy(testFunc, {
    apply(target, thisArg, args) {
        console.time('messure_time');
        target();
        console.timeEnd('messure_time');
    },
});

proxy();
// done
// messure_time: 4.525s

```



proxy.js

```javascript
const originalObject = {
  wheels: 4,
  speed: 100,
};

const proxyObject = new Proxy(originalObject, {
  get(originalObject, key, proxy) {
    console.log('We are trying to get a property!');
    return originalObject[key];
  },
  set(originalObject, key, value) {
    console.log('We are trying to set a property!');
    originalObject[key] = value;
    return true;
  },
});

proxyObject.wheels; // We are trying to get a property!
proxyObject.wheels = 2; // We are trying to set a property!

console.log(originalObject); // { wheels: 2, speed: 100 }

```

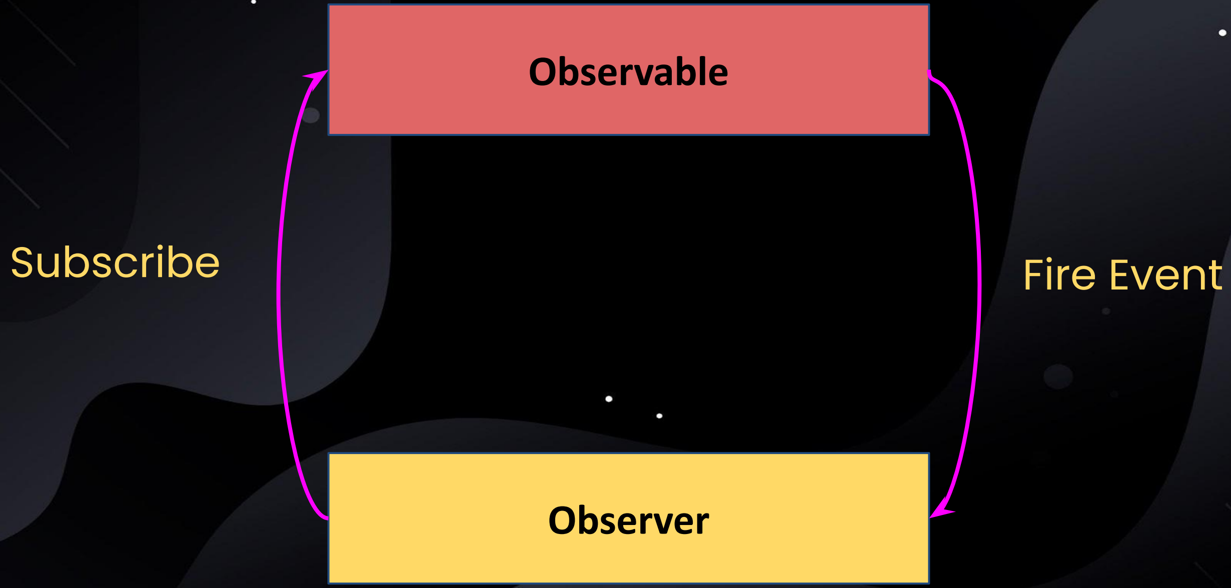


```
<html>
  <body>
    <h1>Count <span id="value"></span></h1>
    <button id="increment">Increment</button>
    <script>
      const valueElement = document.querySelector('#value');
      valueElement.innerText = "0";

      const incrementElement = document.querySelector('#increment');

      const state = new Proxy(
        {value: 0},
        {
          set(object, key, value) {
            valueElement.innerText = value;
            valueElement.style.color = +value % 2 ? 'red' : 'blue';
            object[key] = value;
            return true;
          },
        }
      );

      incrementElement.addEventListener('click', () => {
        state.value++;
      });
    </script>
  </body>
</html>

```

Observer

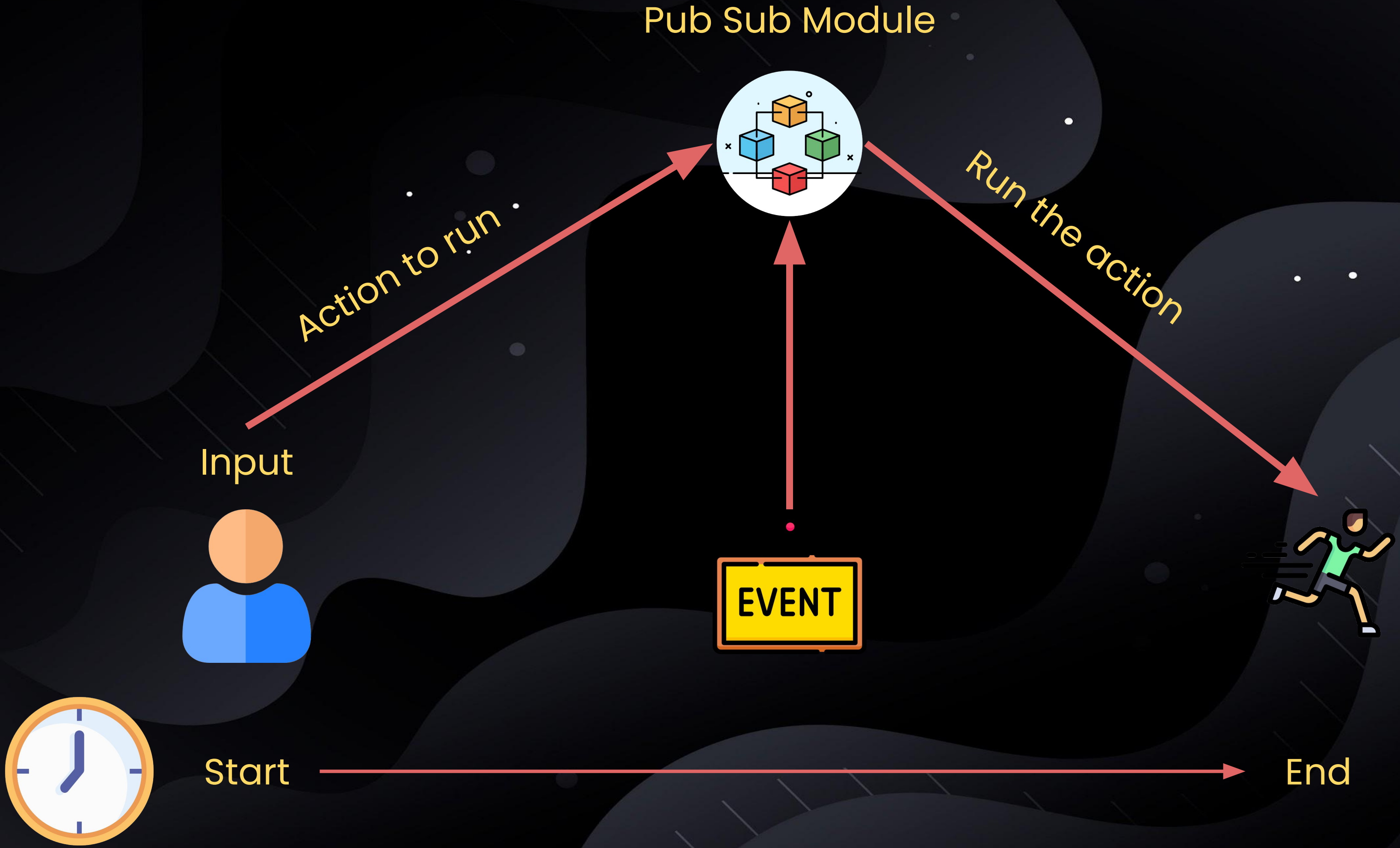
- A pattern allowing to subscribe object (Observers) to another object (Observable/Subject)
- When an event is triggered Observable notifies all its Observers

Observer

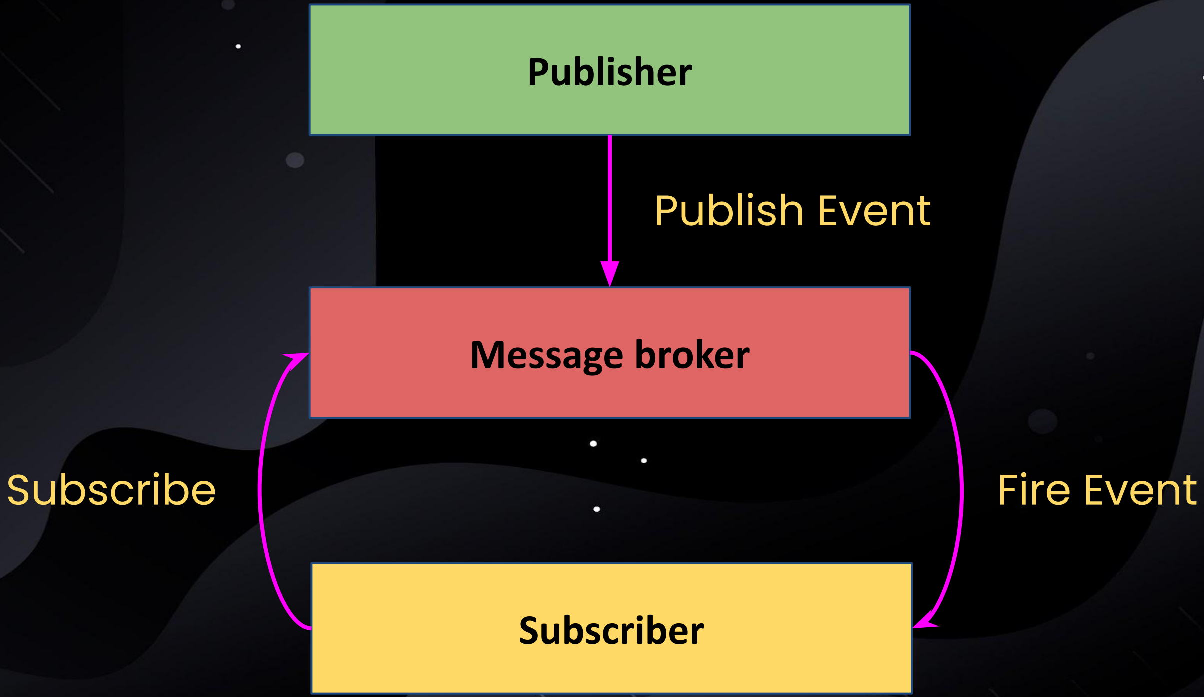


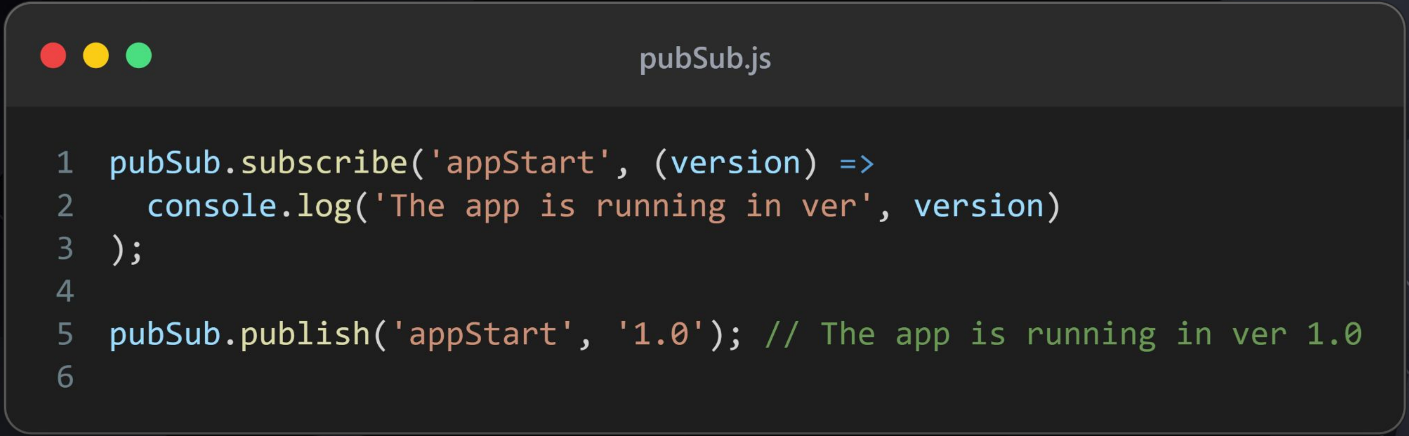
Pub Sub

- A module that allows to subscribe to an event
- When an event happens the module will run your code
- Way to orchestrate work in your code and decoupling of objects
- Very popular - event in DOM, Node.js, Electron

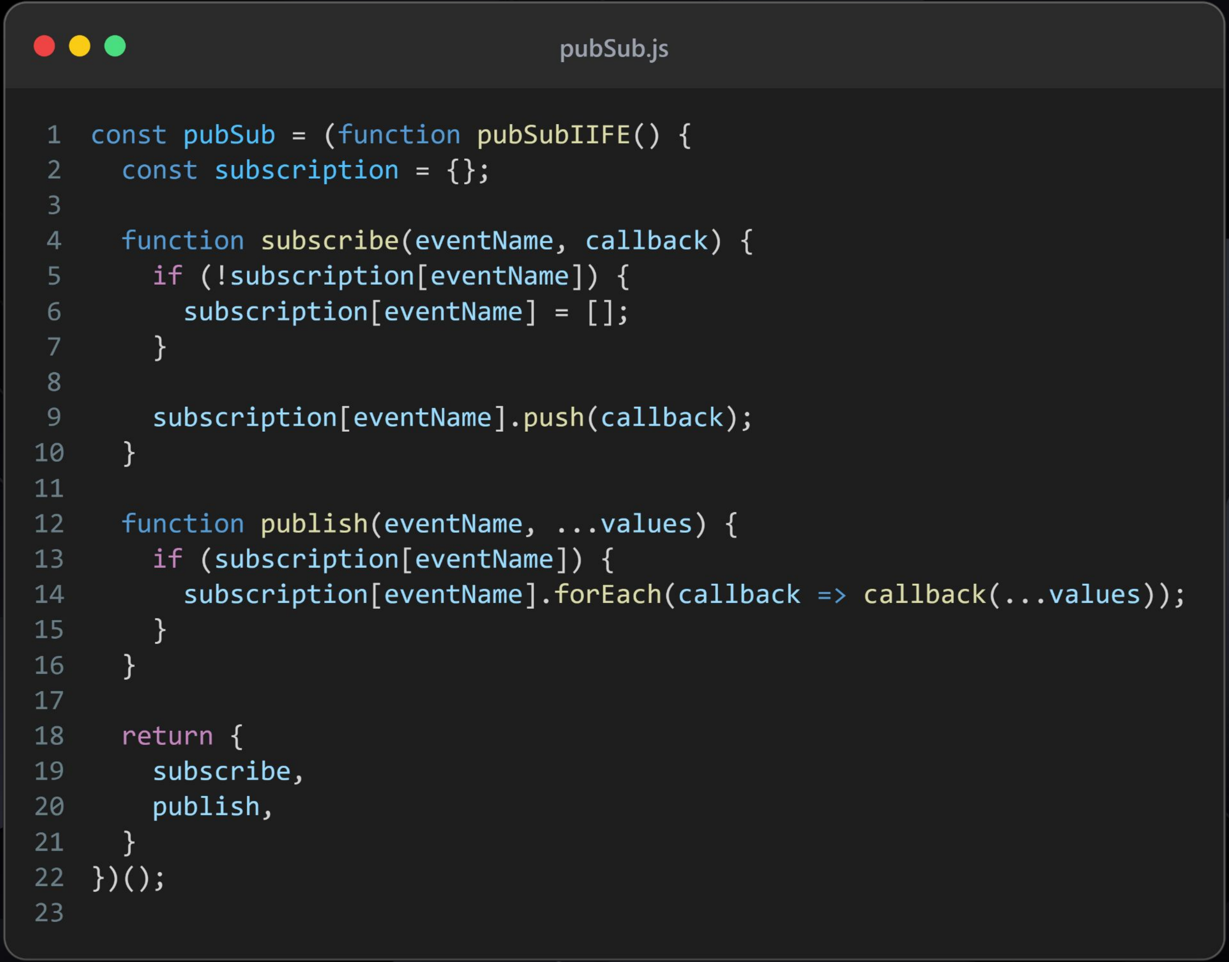

Pub Sub Module
Action to run
Run the action
Input
EVENT
Start
End

Pub Sub

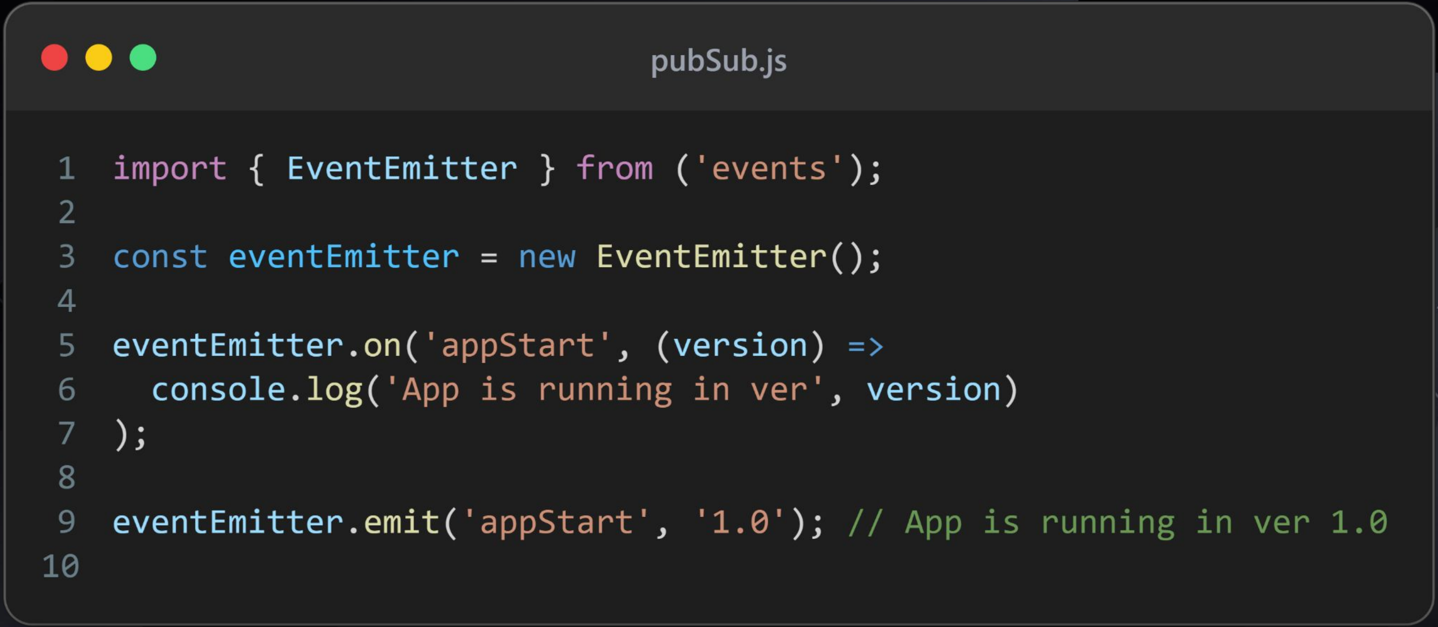

Publisher
Publish Event
Message broker
Subscribe
Fire Event
Subscriber


pubSub.js

```javascript
pubSub.subscribe('appStart', (version) =>
  console.log('The app is running in ver', version)
);

pubSub.publish('appStart', '1.0'); // The app is running in ver 1.0
```

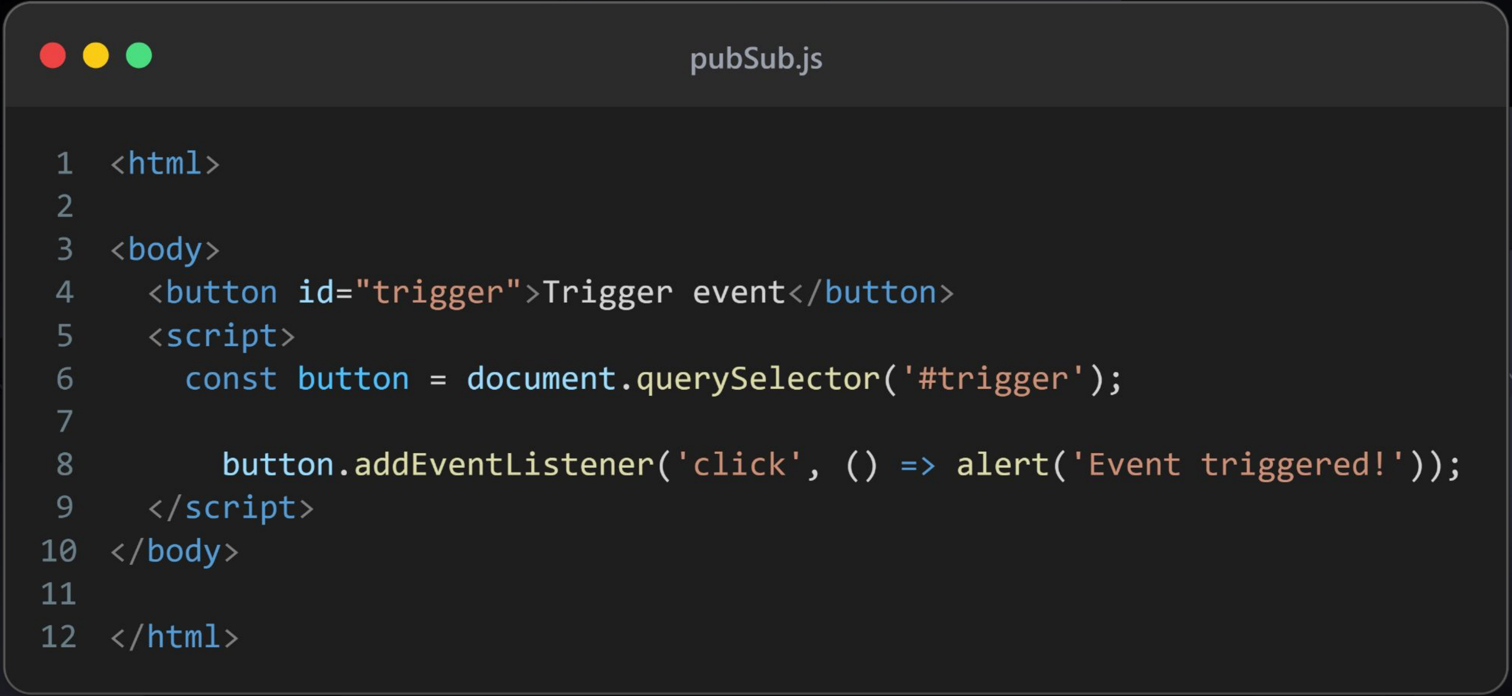

pubSub.js

```
const pubSub = (function pubSubIIFE() {
  const subscription = {};

  function subscribe(eventName, callback) {
    if (!subscription[eventName]) {
      subscription[eventName] = [];
    }

    subscription[eventName].push(callback);
  }

  function publish(eventName, ...values) {
    if (subscription[eventName]) {
      subscription[eventName].forEach(callback => callback(...values));
    }
  }

  return {
    subscribe,
    publish,
  }
})();

```



pubSub.js

```javascript
import { EventEmitter } from ('events');

const eventEmitter = new EventEmitter();

eventEmitter.on('appStart', (version) =>
  console.log('App is running in ver', version)
);

eventEmitter.emit('appStart', '1.0'); // App is running in ver 1.0
```

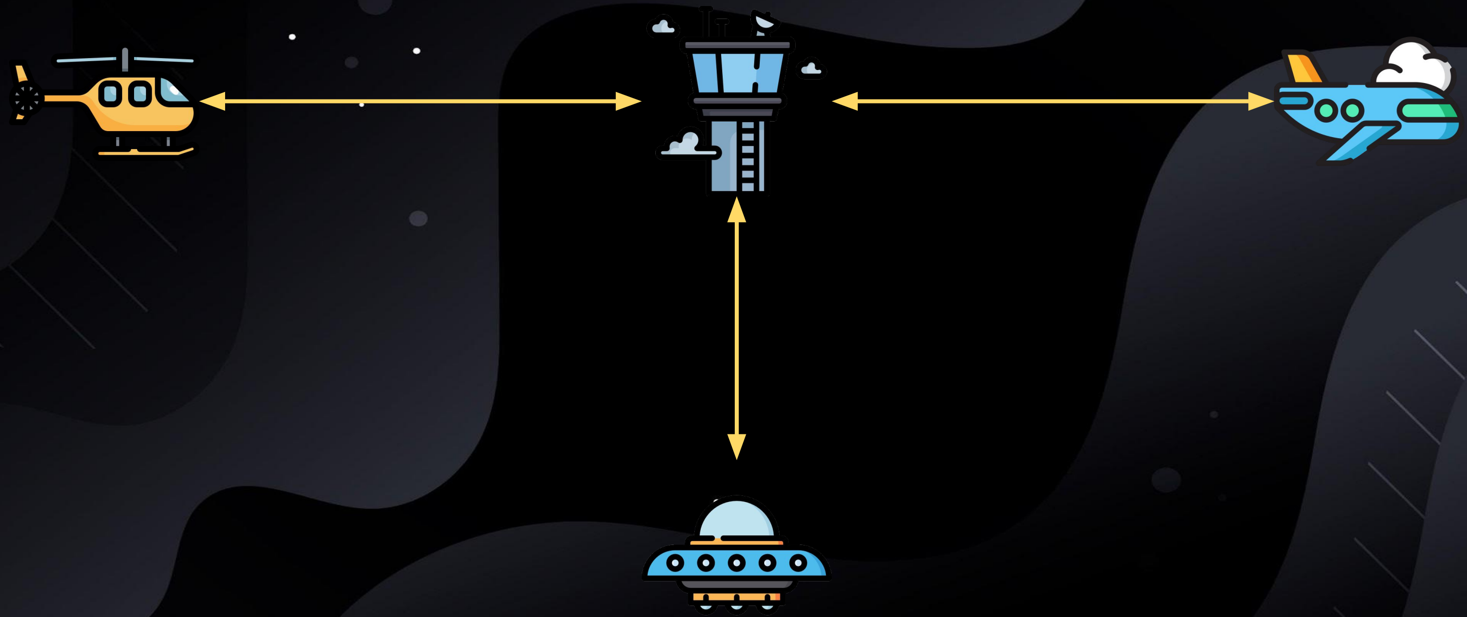

pubSub.js

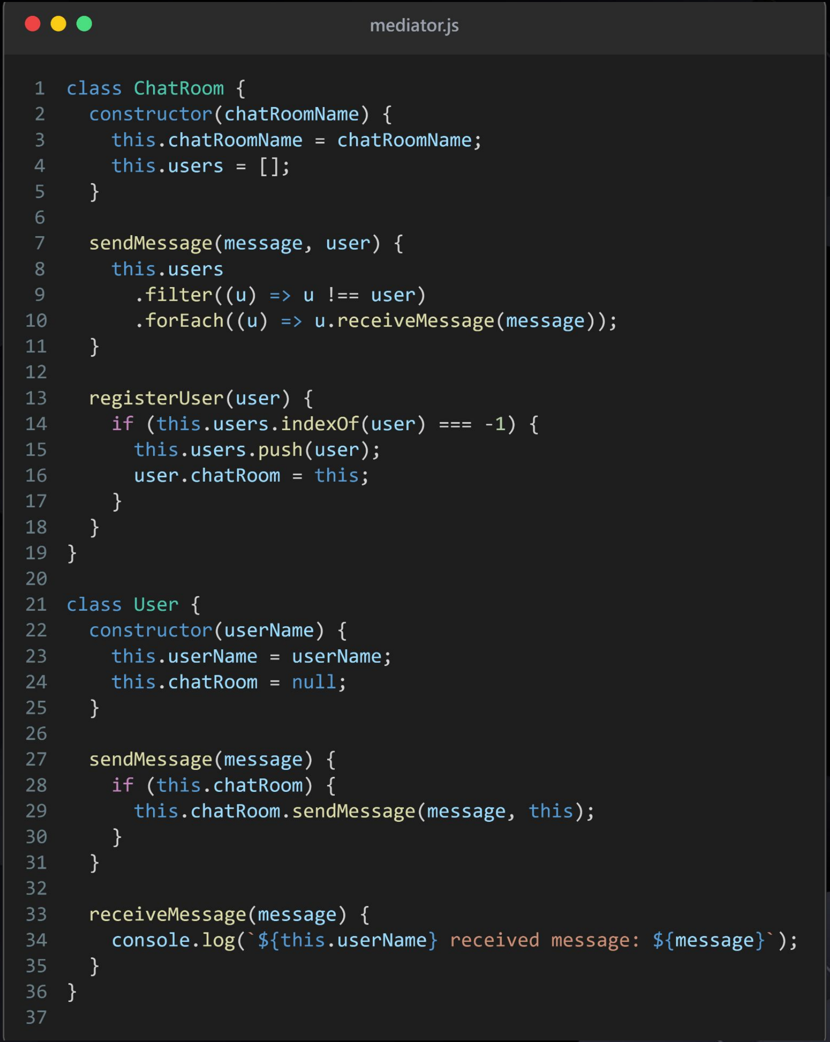
```
<html>

<body>
  <button id="trigger">Trigger event</button>
  <script>
    const button = document.querySelector('#trigger');

      button.addEventListener('click', () => alert('Event triggered!'));
  </script>
</body>

</html>
```

Mediator

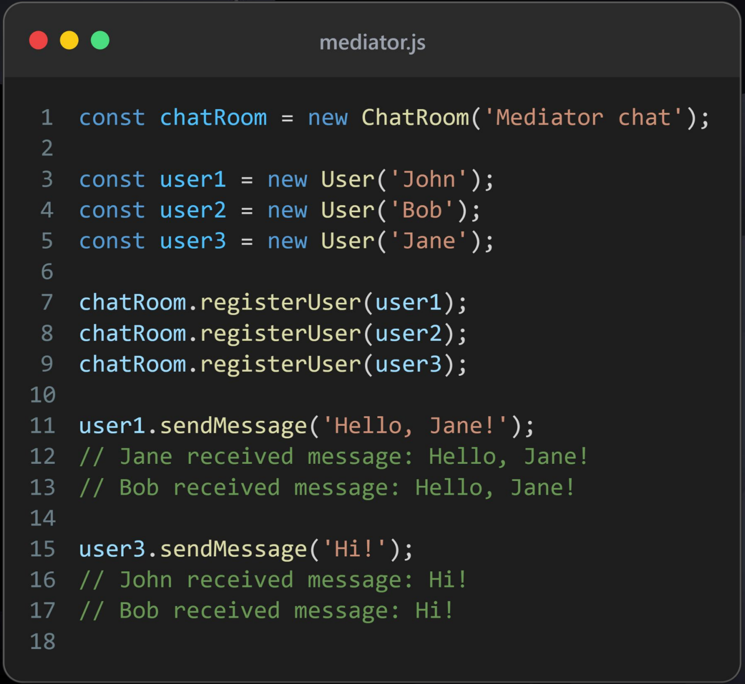
- Extends the concepts of proxy and pub-sub to orchestrate events more granularly
- Solves the problem of many too many interactions

Mediator Module



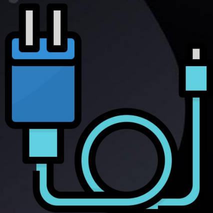

mediator.js

```javascript
class ChatRoom {
  constructor(chatRoomName) {
    this.chatRoomName = chatRoomName;
    this.users = [];
  }

  sendMessage(message, user) {
    this.users
      .filter((u) => u !== user)
      .forEach((u) => u.receiveMessage(message));
  }

  registerUser(user) {
    if (this.users.indexOf(user) === -1) {
      this.users.push(user);
      user.chatRoom = this;
    }
  }
}

class User {
  constructor(userName) {
    this.userName = userName;
    this.chatRoom = null;
  }

  sendMessage(message) {
    if (this.chatRoom) {
      this.chatRoom.sendMessage(message, this);
    }
  }

  receiveMessage(message) {
    console.log(`${this.userName} received message: ${message}`);
  }
}

```


mediator.js

```javascript
const chatRoom = new ChatRoom('Mediator chat');

const user1 = new User('John');
const user2 = new User('Bob');
const user3 = new User('Jane');

chatRoom.registerUser(user1);
chatRoom.registerUser(user2);
chatRoom.registerUser(user3);

user1.sendMessage('Hello, Jane!');
// Jane received message: Hello, Jane!
// Bob received message: Hello, Jane!

user3.sendMessage('Hi!');
// John received message: Hi!
// Bob received message: Hi!

```

Adapter

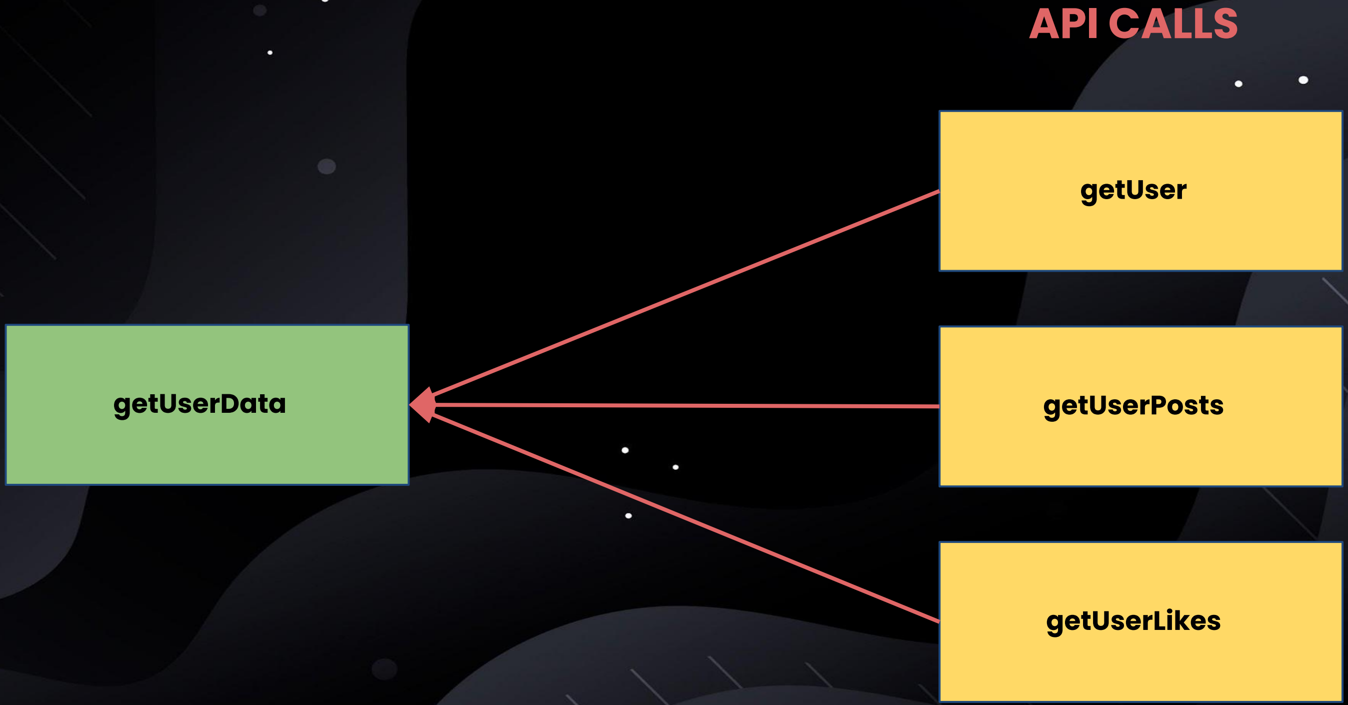


- Wrapper usually for dependency that exists as the signal source of truth
- Solves the problem of having to change an external dependency
- Intent of Adapter is to design to an existing interface

Facade

- Wrapper usually for complicated subsystem
- Solves the problem of complexity by providing simpler interface

Facade


API CALLS
getUserData
getUser
getUserPosts
getUserLikes

THE
END