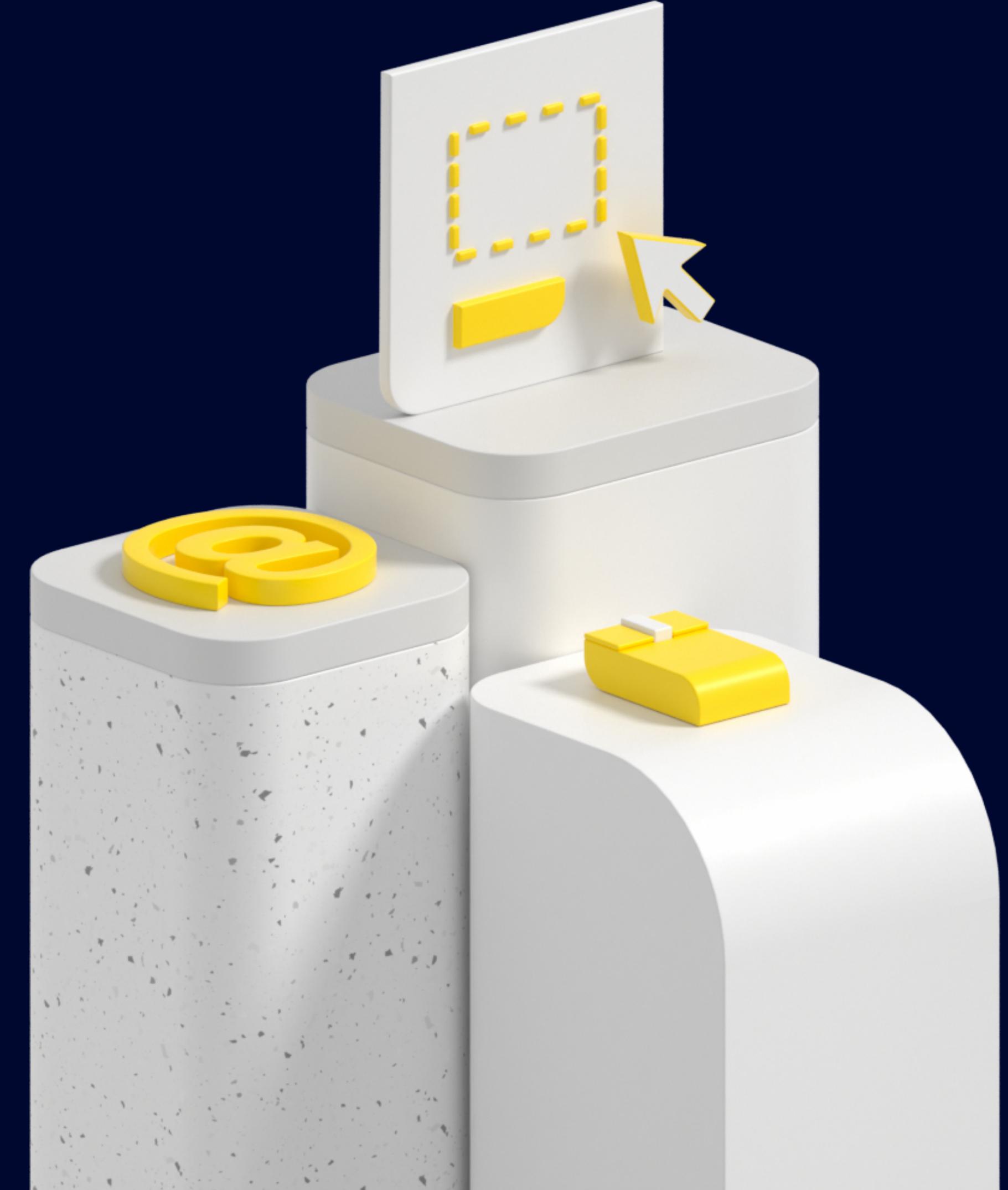




DOM, БРАУЗЕРНЫЕ СОБЫТИЯ

23.03.2023





Роман Кузьменко | Фронтендер в Т-Ж



[@kuzmrom](https://t.me/kuzmrom)

СЕГОДНЯ:

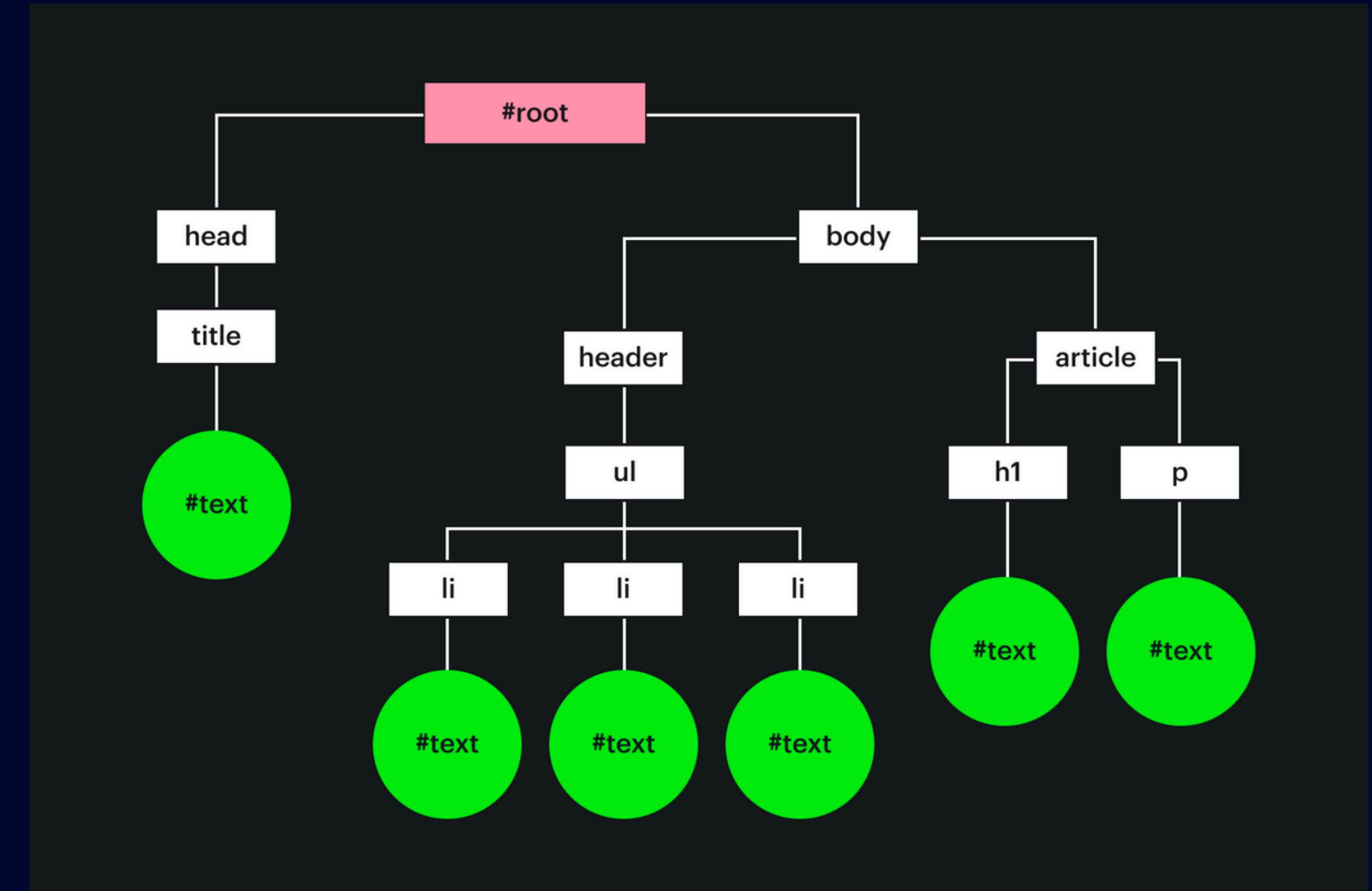
- DOM
- Манипуляция DOM-элементами
- События DOM
- Пользовательские элементы

DOM – ОБЪЕКТНАЯ МОДЕЛЬ ДОКУМЕНТА

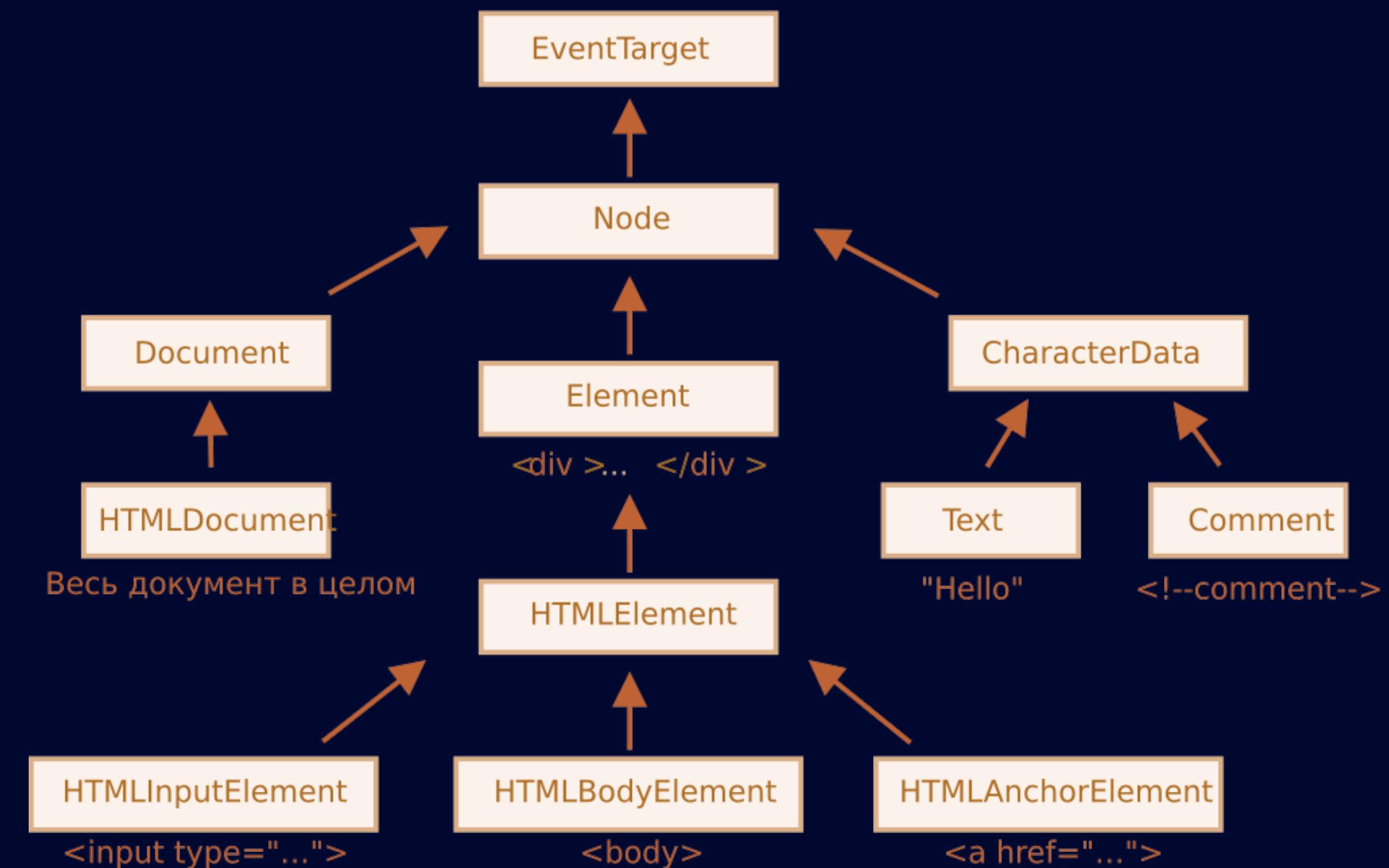
1. Структурированное представление документа
2. API для работы с ним (навигация, поиск, изменение)



```
<!DOCTYPE html>
<head>
  <title>Личный кабинет</title>
</head>
<body>
  <header>
    <ul class="menu">
      <li>Главная</li>
      <li>Статьи</li>
      <li>Контакты</li>
    </ul>
  </header>
  <article id="12">
    <h1>Как выучить джаваскрипт?</h1>
    <p>Нужно начать учиться.</p>
  </article>
</body>
```



УЗЛЫ DOM



API для работы с узлами

- Навигация
- Поиск
- Изменение

Document

- childNodes => NodeList
- children => HTMLCollection

МАССИВОПОДОБНЫЕ ОБЪЕКТЫ

- Не являются массивами (нет методов массивов)
- Являются итерируемыми (можно перебрать с помощью цикла `for of`, использовать вместе со `spread` и `rest`, и т.д.)
- Динамические (отслеживают изменения)

[MDN: HTMLCollection](#) [MDN: NodeList](#)

НАВИГАЦИЯ ПО ЭЛЕМЕНТАМ



```
document.documentElement; // html-элемент  
document.body; // body-элемент  
element.parentElement; // Родитель  
element.childNodes; // NodeList  
element.children; // HTMLCollection  
  
element.firstChild; element.firstElementChild;  
element.lastChild; element.lastElementChild;  
  
element.previousElementSibling;  
element.nextElementSibling;
```

ПОИСК ЭЛЕМЕНТОВ

```
// Поиск по ID, name
document.getElementById('elementId'); // DOM элемент или null
document.getElementsByName('name'); // NodeList

// Поиск по class, тегу, есть у всех элементов
element.getElementsByClassName('class'); // HTMLCollection
element.getElementsByTagName('div'); // HTMLCollection

// Поиск по css-селектору
element.querySelector('div > div'); // Первый DOM элемент
element.querySelectorAll('div'); // NodeList (СТАТИЧЕСКИЙ!)
```

СОДЕРЖИМОЕ ЭЛЕМЕНТА



```
<div id="foo" style="text-transform: uppercase;">
  <span>bar</span>, <span>baz</span>
</div>
<script>
  const div = document.getElementById('foo');
  div.textContent; // 'bar, baz'
  div.innerText; // 'BAR, BAZ' - нестандартное свойство
  div.innerHTML; // '<span>bar</span>, <span>baz</span>'
</script>
```

АТРИБУТЫ ЭЛЕМЕНТА

```
● ● ●  
    id="email-input"  
    type="email"  
    name="usermail"  
    placeholder="Электронная почта"  
/>  
<script>  
  const input = document.getElementById('email-input');  
  input.setAttribute('hidden', true);  
  input.setAttribute('placeholder', 'E-Mail');  
  input.getAttribute('name'); // 'usermail'  
</script>
```

DATA-АТРИБУТЫ ЭЛЕМЕНТА



```
<div class="user" data-id="0">Петя</div>
<div class="user" data-id="1">Маша</div>
<div class="user" data-id="239">Вася</div>

<script>
  const element = document.querySelector('[data-id="1"]');
  element.setAttribute('data-age', 22);
  element.dataset.age; // 22
  element.dataset.id; // 1
  element.dataset.age = 33;
</script>
```

СОЗДАНИЕ И ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ

```
<div id="parent"></div>

<script>
  const parent = document.getElementById('parent');

  const paragraph = document.createElement('p');
  paragraph.textContent = 'Foo';

  parent.appendChild(paragraph);
  parent.textContent; // 'Foo'

  const paragraph2 = paragraph.cloneNode();
  paragraph2.textContent = 'Bar';

  parent.insertBefore(paragraph2, paragraph);
  parent.textContent; // 'BarFoo'
</script>
```

УДАЛЕНИЕ ЭЛЕМЕНТОВ

```
<div id="parent">  
  <p>Foo</p>  
</div>  
  
<script>  
  const parent = document.getElementById('parent');  
  parent.removeChild(parent.firstChild);  
  // OR  
  parent.firstChild.remove()  
</script>
```

СОБЫТИЯ

— это сигнал от браузера о том, что что-то произошло. Все DOM-узлы подают такие сигналы.



```
const button = document.querySelector('.btn');
button.addEventListener('click', event => { ... });

document.addEventListener('DOMContentLoaded', event => { ... });
```

САМЫЕ РАСПРОСТРАНЕННЫЕ:

- События мыши (click, mousemove)
- События клавиатуры (keyup, keydown)
- События окна (scroll, resize)
- События формы (input, change, focus, submit)

ОТМЕНА ДЕЙСТВИЙ ПО-УМОЛЧАНИЮ



```
form.addEventListener('submit', event =>
{ // Предотвратит отправку запроса
  event.preventDefault();
})
link.addEventListener('click', event => {
  // Предотвратит переход по ссылке
  event.preventDefault();
})
```

УДАЛЕНИЕ ОБРАБОТЧИКА СОБЫТИЙ



```
function handleMouseClick(event) {
    console.log('Вы нажали на элемент:', event.target)
}

// Добавляем обработчик события
document.addEventListener('click', handleMouseClick)

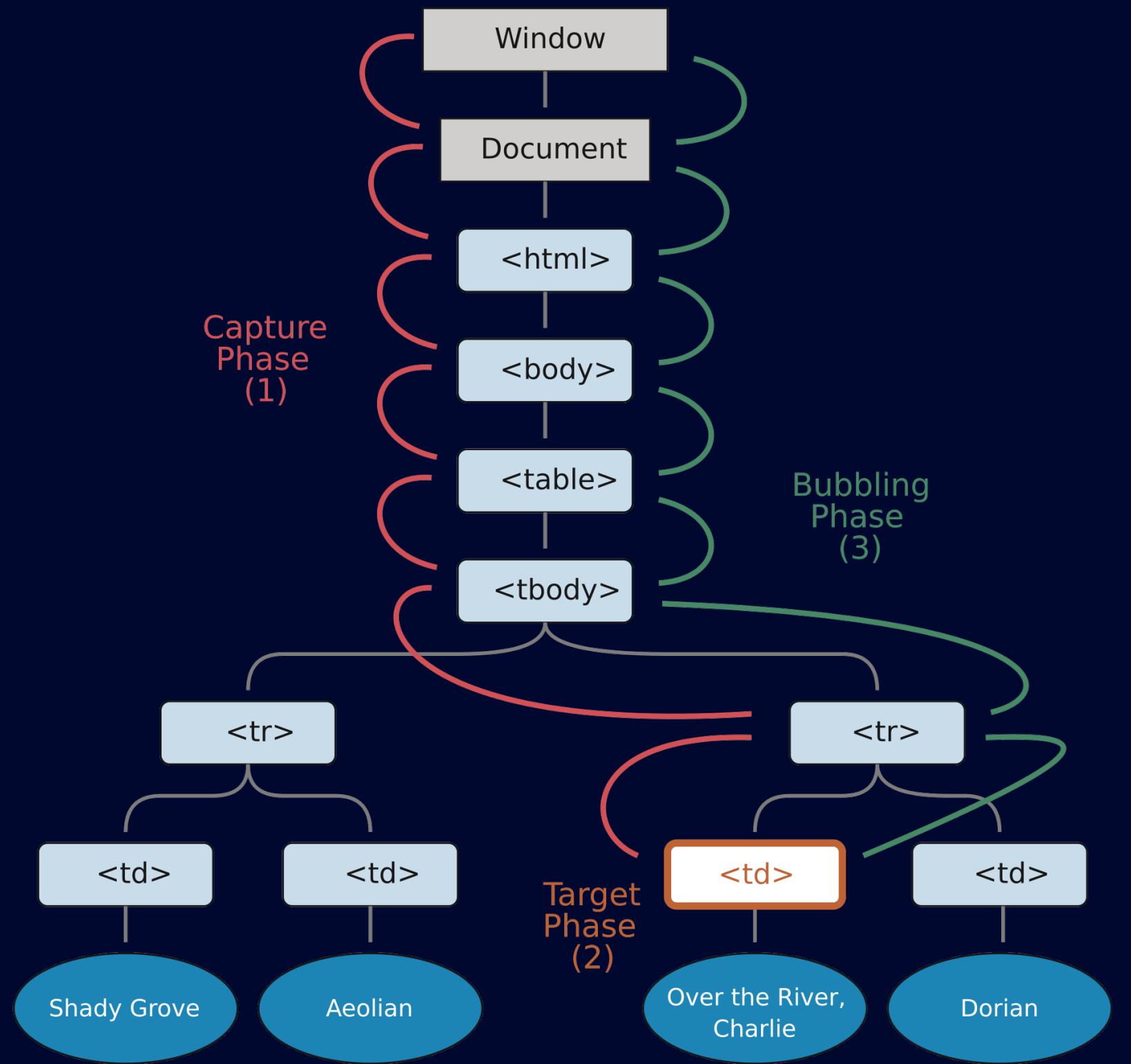
// Убираем обработчик события
document.removeEventListener('click',
handleMouseClick)
```

[Doka: removeEventListener](#)

ВСПЛЫТИЕ И ПЕРЕХВАТ СОБЫТИЙ

Две **Три** фазы распространения события:

1. Capture
2. Target
3. Bubbling



ШАБЛОНИЗАЦИЯ

```
const movie = {  
    title: 'Interstellar',  
    year: 2015,  
    imdbId: 'tt0816692',  
    poster: 'https://example.com/interstellar.jpg'  
};
```



```
<div class="movie">  
  <a class="movie-link" href="https://www.imdb.com/title/tt0816692/">  
    <h3 class="movie-title">Interstellar (2015)</h3>  
      
  </a>  
</div>
```

ШАБЛОНIZАЦИЯ: В ЛОБ

```
const render = movieData => {
  const movie = document.createElement("div");
  movie.classList.add("movie");

  const link = document.createElement("a");
  link.classList.add("movie-link");
  link.setAttribute("target", "_blank");
  link.setAttribute("href", movieData.link);
  movie.appendChild(link);

  const title = document.createElement("h3");
  title.classList.add("movie-title");
  title.textContent = `${movieData.title} (${movieData.year})`;

  const poster = document.createElement("img");
  poster.classList.add("movie-poster");
  poster.setAttribute("src", movieData.poster);
  link.appendChild(title);
  link.appendChild(poster);

  return movie;
};

document.body.appendChild(render(movie));
```

ШАБЛОНИЗАЦИЯ: innerHTML

```
● ● ●  
const render = movieData => {  
  const movie = document.createElement('div');  
  movie.classList.add('movie');  
  movie.innerHTML = `  
    <a class="movie-link" href="${movieData.link}">  
      <h3 class="movie-title">${movieData.title} (${movieData.year})</h3>  
        
    </a> `;  
  return movie;  
};  
  
document.body.appendChild(render(movie));
```

ШАБЛОНИЗАЦИЯ: template

```
● ● ●

<template id="movie">
  <div class="movie">
    <a class="movie-link" href="">
      <h3 class="movie-title"></h3>
      
    </a>
  </div>
</template>

<script>
  const render = movieData => {
    const movie = document.getElementById('movie').content.cloneNode(true);
    const link = movie.querySelector('.movie-link');
    const title = movie.querySelector('.movie-title');
    const poster = movie.querySelector('.movie-poster');
    link.href = movieData.link;
    title.textContent = `${movieData.title} (${movieData.year})`;
    poster.src = movieData.poster;
    return movie;
  };
  document.body.appendChild(render(movie));
</script>
```

ВЕБ-КОМПОНЕНТЫ:

- ES Modules
- HTML Templates
- Custom Elements
- Shadow DOM

CUSTOM ELEMENTS

```
● ● ●  
class CurrentYear extends HTMLElement {  
    constructor() {  
        super();  
        this.textContent = `${new Date().getFullYear()}`;  
    }  
}  
customElements.define('current-year', CurrentYear);  
// html  
// <current-year></current-year>
```

SHADOW DOM

```
● ● ●

class CurrentYear extends HTMLElement {
  constructor() {
    super();
    const shadow = this.attachShadow({ mode: 'open' });

    shadow.innerHTML = ` ${new Date().getFullYear()} `;
  }
}
customElements.define('current-year', CurrentYear);
```

Жизненный цикл HTMLElement

```
class MyElement extends HTMLElement {  
    constructor() {  
        super();  
        // ...  
    }  
    connectedCallback() {  
        // При добавлении в DOM  
    }  
    disconnectedCallback() { // При удалении из DOM  
    }  
    attributeChangedCallback(name, oldValue, newValue) { // При изменении атрибута  
    }  
    adoptedCallback() {  
        // При переносе элемента в другой документ  
    }  
}
```

