Frontender[1.0] JavaScript - События (events). Погружение, всплытие и делегирование событий

YouTube	https://youtu.be/8jemKgy1CIM
Telegram	https://t.me/Dmitry_Kolotilshikov
Github	https://github.com/DmitryKolotilshikov/
Boosty	https://boosty.to/dmitry_ko
# Номер урока	34

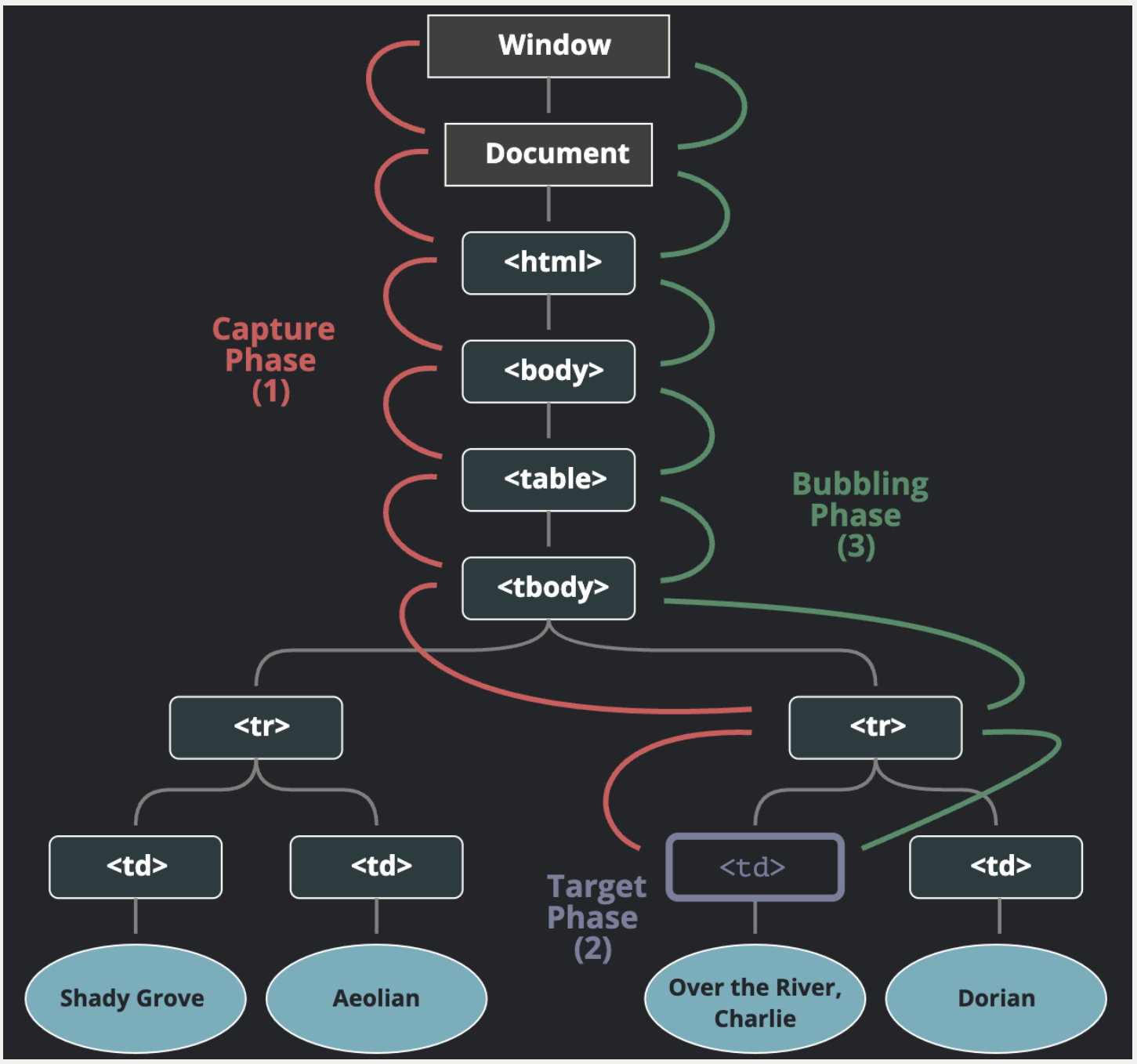
💡 Задание к этому уроку - **закрепить данный урок руками**

💡 Полезные ссылки:

https://learn.javascript.ru/bubbling-and-capturing

https://learn.javascript.ru/event-delegation

💡 Схема погружения (capture) и всплытия (bubbling) события



💡 Зачем нужно **делегирование событий**?

Зачем использовать:

- Упрощает процесс инициализации и экономит память: не нужно вешать много обработчиков.
- Меньше кода: при добавлении и удалении элементов не нужно ставить или снимать обработчики.
- Удобство изменений DOM: можно массово добавлять или удалять элементы путём изменения `innerHTML` и ему подобных.

Конечно, у делегирования событий есть свои ограничения:

- Во-первых, событие должно всплывать. Некоторые события этого не делают. Также, низкоуровневые обработчики не должны вызывать `event.stopPropagation()`.
- Во-вторых, делегирование создаёт дополнительную нагрузку на браузер, ведь обработчик запускается, когда событие происходит в любом месте контейнера, не обязательно на элементах, которые нам интересны. Но обычно эта нагрузка настолько пустяковая, что её даже не стоит принимать во внимание.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js" defer></script>
    <title>Events - propagation and delegation</title>
    <style>
      .borders {
        padding: 1em;
        border: 1px solid orangered;
      }
      .borders p {
        border: 1px solid lightseagreen;
      }
      table {
        width: max-content;
        border-collapse: collapse;
      }
      td {
        padding: 1em;
        width: 80px;
        height: 80px;
        border: 1px solid lightseagreen;
      }
      .highlight {
        background-color: cadetblue;
      }
```

```
    </style>
</head>

<body>
  <div data-container class="borders">
    <p>кликните <strong>тут</strong> и сработает обработчик на DIV</p>
  </div>
  <hr>
  <table data-numpad>
    <tr>
      <td>7</td>
      <td>8</td>
      <td>9</td>
    </tr>
    <tr>
      <td>4</td>
      <td>5</td>
      <td>6</td>
    </tr>
    <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
  </table>
  <hr>
  <section data-table></section>
</body>

</html>
```

```
// Events - propagation & delegation
// 1 - Погружение и всплытие событий (events capture and bubbling)
// 2- Делегирование событий (events delegation)

const log = console.log;

// 💎 --- Погружение и всплытие событий (events capturing and bubbling) --- 💎
/*
 - Всплытие -
Когда на элементе происходит событие, обработчики сначала срабатывают на
нём, потом на его родителе, затем выше и так далее, вверх по цепочке предков.
*/

const container = document.querySelector("[data-container]");

// container.addEventListener("click", () => {
//     log("обработчик контейнера сработал!");
// })

document.addEventListener("click", () => log("клик по всему документу!"));

container.addEventListener("click", (e) => {
    // запрещает всплытие и погружение
    // e.stopPropagation();
    // останавливает обработку событий на этом же элементе, если есть еще события
    // e.stopImmediatePropagation();
```

```
    log("1) обработчик контейнера сработал!");
});

// container.addEventListener("click", () => {
//     log("2) обработчик контейнера сработал!");
// });

/*
Погружение
В современной разработке стадия погружения используется очень редко,
обычно события обрабатываются во время всплытия.

3й аргумент функции addEventListener или просто true или {capture: true}
*/

document.addEventListener("click", () => log("клик по всему документу!"), true);

container.addEventListener("click", () => {
    log("2) обработчик контейнера сработал!");
}, true);


// 💎 --- Делегирование событий --- 💎
/*
Всплытие и погружение являются основой для «делегирования событий» –
очень мощного приёма обработки событий.
*/

const table = document.createElement("table");

for (let row = 0; row < 100; row++) {
    const tr = document.createElement("tr");

    for (let col = 0; col < 50; col++) {
        const td = document.createElement("td");
        td.dataset.cell = `row${row+1}-col${col+1}`;
        td.textContent = td.dataset.cell;
        tr.append(td);
    }

    table.append(tr);
}

table.addEventListener("click", (e) => {
    for (const elem of table.querySelectorAll("*")) {
        elem.classList.remove("highlight")
    }
    if (e.target.tagName === "TD") {
        e.target.classList.add("highlight");
    }
})

document.querySelector("[data-table]").append(table);

// ------------------ numpad ------------------

const numpad = document.querySelector("[data-numpad]");
```

```
document.addEventListener("keypress", (e) => {
    for (const button of numpad.querySelectorAll("td")) {
        if (button.textContent === e.key) {
            button.classList.add("highlight");
        }
    };
})

document.addEventListener("keyup", () => {
    for (const button of numpad.querySelectorAll("td")) {
        button.classList.remove("highlight");
    };
})
```