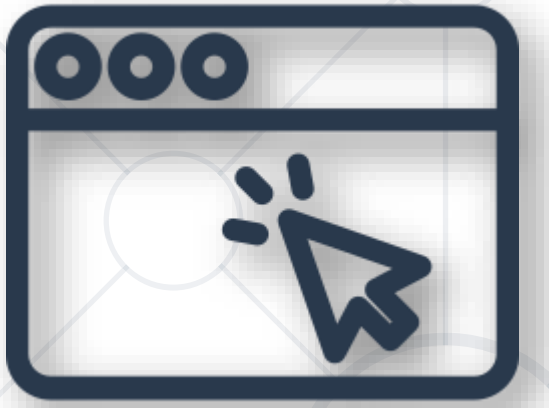
DOM Events

Handling DOM Events, Propagation & Delegation



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

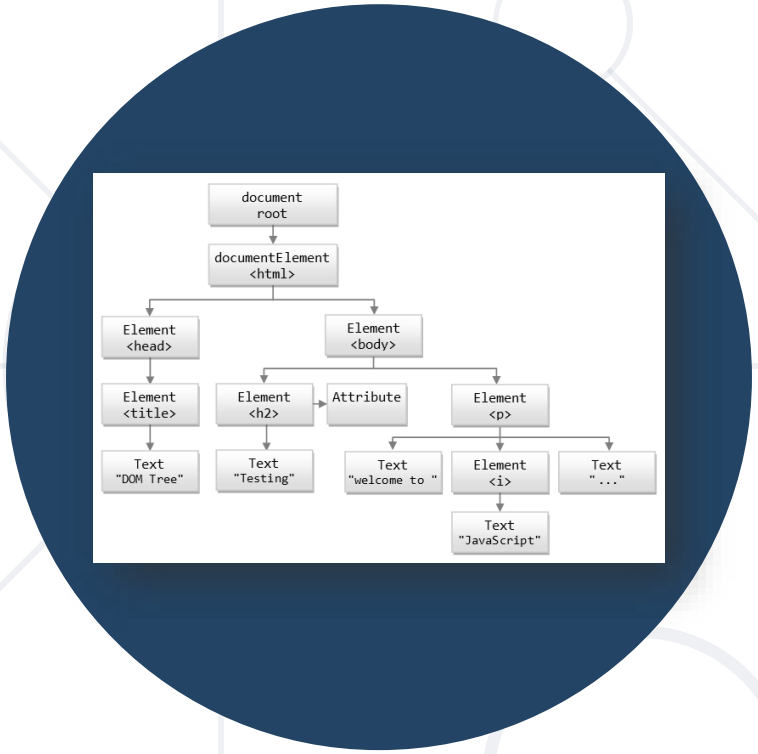
Have a Question?

Software University

sli.do

#js-front-end

Table of Contents

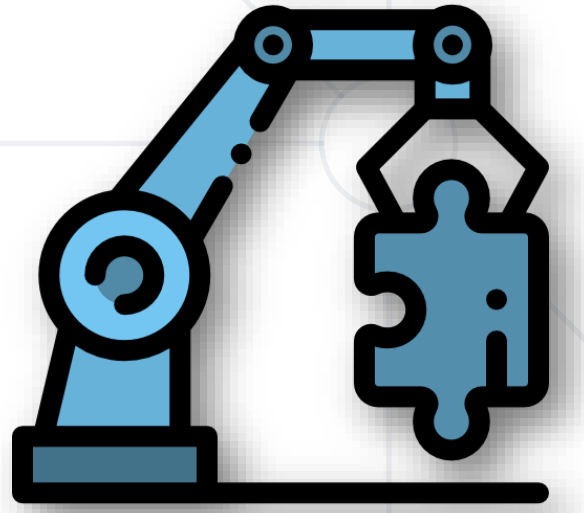


DOM Manipulation

Modify the DOM Tree

DOM Manipulations

- We can **create**, **append** and **remove** HTML elements dynamically
  - `appendChild()`
  - `removeChild()`
  - `replaceChild()`



Creating New DOM Elements

- HTML elements are created with **`document.createElement`**
  - This is called a **Factory Pattern**
- Variables holding HTML elements are **live**:
  - If you **modify** the contents of the variable, the DOM is **updated**
  - If you **insert** it somewhere in the DOM, the original is **moved**
- Text added to **`textContent`** will be **escaped**
- Text added to **`innerHTML`** will be **parsed** and turned into actual HTML elements → beware of **XSS attacks**!

Creating DOM Elements

- Creating a new DOM element

```
let p = document.createElement("p");
let li = document.createElement("li");
```

Tag name

- Create a copy / cloning DOM element

```
let li = document.getElementById("my-list");
let newLi = li.cloneNode(true);
```

- Elements are created **in memory** – they don't exist on the page
- To become visible, they must be **appended** to the DOM tree

Manipulating Node Hierarchy

- `appendChild` - Adds a new child, as the **last child**

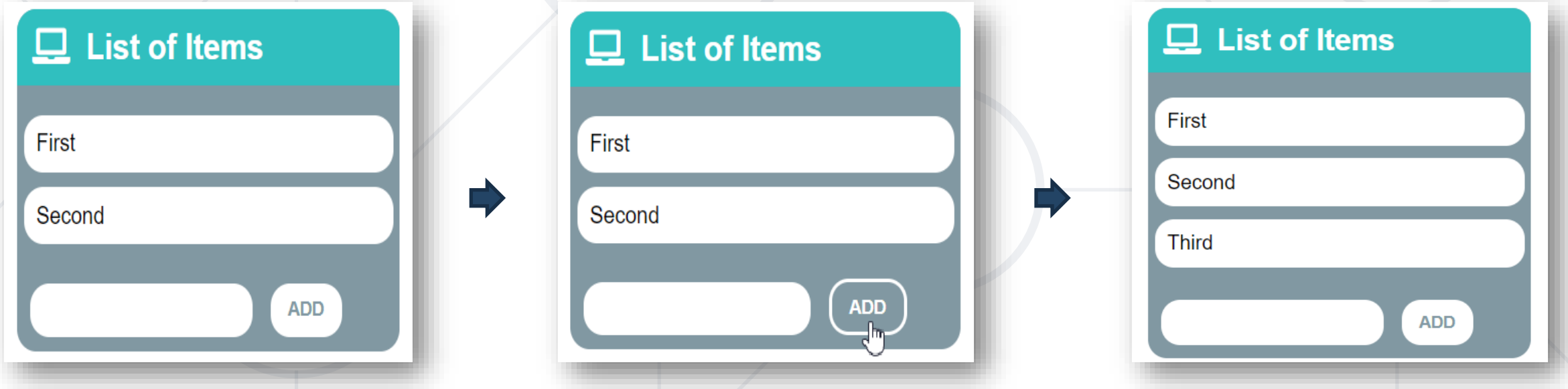
```
let p = document.createElement("p");
let li = document.createElement("li");
li.appendChild(p);
```

- `prepend` - Adds a new child, as the **first child**

```
let ul = document.getElementById("my-list");
let li = document.createElement("li");
ul.prepend(li);
```

Problem: List of Items

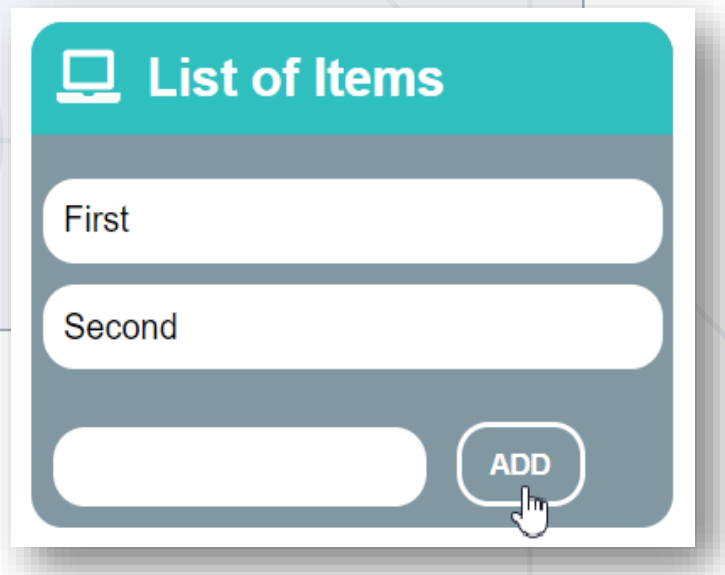
- Create a HTML page holding a **list of items** + **text box** + **button** for adding more items to the list
  - Write a function to **append** the specified text to the list



Check your solution here: https://judge.softuni.org/Contests/Practice/Index/4367#0

Problem: List of Items – HTML

```
<h1>List of Items</h1>
<ul id="items"><li>First</li><li>Second</li></ul>
<input type="text" id="newItemText" />
<input type="button" value="Add" onclick="addItem()">
<script>
function addItem() {
  // TODO: Add new item to the list
}
</script>
```



Solution: List of Items

```
function addItem() {
  let text = document.getElementById('newItemText').value;
  let li = document.createElement("li");
  li.appendChild(document.createTextNode(text));
  document.getElementById("items").appendChild(li);
  // Clearing the input
  document.getElementById('newItemText').value = '';
}
```

Deleting DOM Elements

Software University

```
<ul id="items">
  <li class="red">Red</li>
  <li class="blue">Blue</li>
</ul>
```

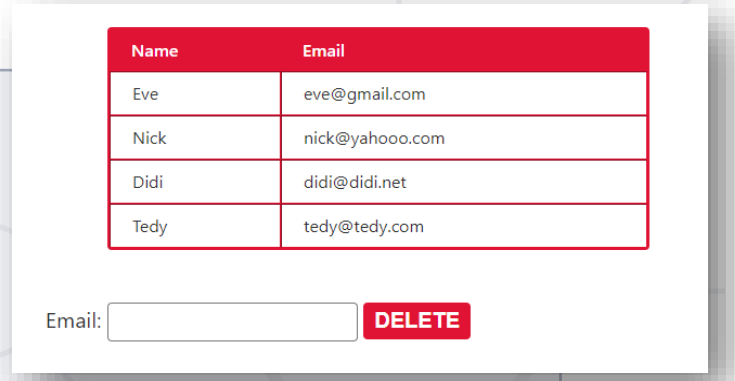
```
▼<body>
  ▼<ul id="items">
      <li class="red">Red</li>
      <li class="blue">Blue</li>
    </ul>
  </body>
```

```
let redElements =
  document.querySelectorAll("#items li.red");
redElements.forEach(li => {
  li.parentNode.removeChild(li);
});
```

```
▼<body>
  ▼<ul id="items">
      <li class="blue">Blue</li>
    </ul>
  </body>
```

Problem: Delete from Table

```
<table border="1" id="customers">
 <tr><th>Name</th><th>Email</th></tr>
 <tr><td>Eve</td><td>eve@gmail.com</td></tr>
 <tr><td>Nick</td><td>nick@yahooo.com</td></tr>
 <tr><td>Didi</td><td>didi@didi.net</td></tr>
 <tr><td>Tedy</td><td>tedy@tedy.com</td></tr>
</table>
Email: <input type="text" name="email" />
<button onclick="deleteByEmail()">Delete</button>
<div id="result" />
```

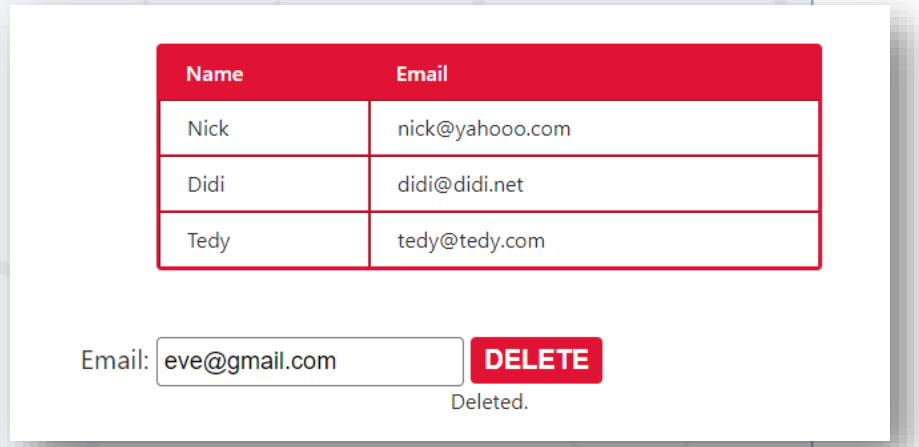


Check your solution here: *https://judge.softuni.org/Contests/Practice/Index/4367#1*

Solution: Delete from Table

Software University

```
function deleteByEmail() {
  let email = document.getElementsByName("email")[0].value;
  let secondColumn = document.querySelectorAll(
    "#customers tr td:nth-child(2)");
  for (let td of secondColumn)
    if (td.textContent == email) {
      let row = td.parentNode;
      row.parentNode.removeChild(row);
      document.getElementById('result').
        textContent = "Deleted.";
      return;
    }
  document.getElementById('result').textContent = "Not found.";
}
```



The DOM Event

Event Object and Types

Event Object

- Calls its **associated function**
- Passes a **single argument** to the function - a **reference** to the event object
- Contains **properties** that describe the event
  - Which **element** triggered the event
  - Screen **coordinates** where it occurred
  - What is the **type** of the event
  - And more

Event Types in DOM API

- **Mouse** events

```
click
mouseover
mouseout
mousedown
mouseup
```

- **Touch** events

```
touchstart
touchend
touchmove
touchcancel
```

- **DOM / UI** events

```
load
unload
resize
dragstart / drop
```

- **Keyboard** events

```
keydown
Keypress
keyup
```

- **Focus** events

```
focus (got focus)
blur (lost focus)
```

- **Form** events

```
input
change
submit
reset
```

Event Handling

Event Handler

- Event registration is done by providing a **callback function**
- Three ways to register for an event:
  - With **HTML Attributes**
  - Using **DOM element properties**
  - Using **DOM event handler** – preferred method

```
function handler(event){
    // this --> object, html reference
    // event --> object, event configuration
}
```

Event Listener

- addEventListener();

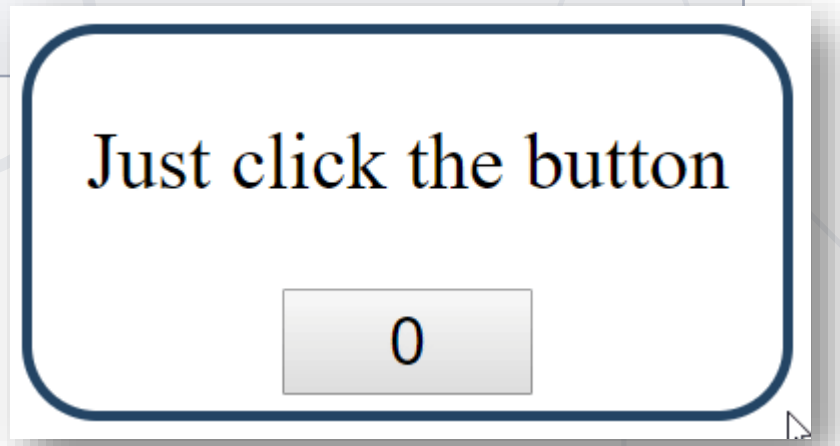
```
htmlRef.addEventListener( 'click' , handler );
```

- removeEventListener();

```
htmlRef.removeEventListener( 'click' , handler);
```

Attaching Click Handler

```
const button = document.getElementsByTagName('button')[0];

button.addEventListener('click', clickMe);

function clickMe(e) {
  const target = e.currentTarget;
  const targetText = target.textContent;
  target.textContent = Number(targetText) + 1;
}
```



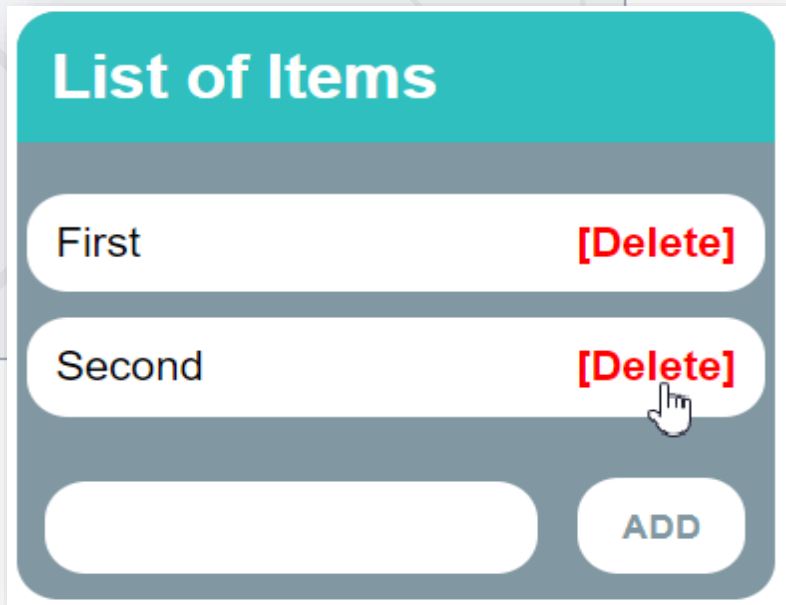
Problem: Add / Delete Items

- Extend the previous problem
  - Implement `[Delete]` action as link after each list item

Check your solution here: *https://judge.softuni.org/Contests/Practice/Index/4367#2*

Problem: Add / Delete Items – HTML

```
<h1>List of Items</h1>
<ul id="items"></ul>
<input type="text" id="newText" />
<input type="button" value="Add" onclick="solve()">
<script>
function solve() {
    // TODO...
}
</script>
```



Solution: Add / Delete Items

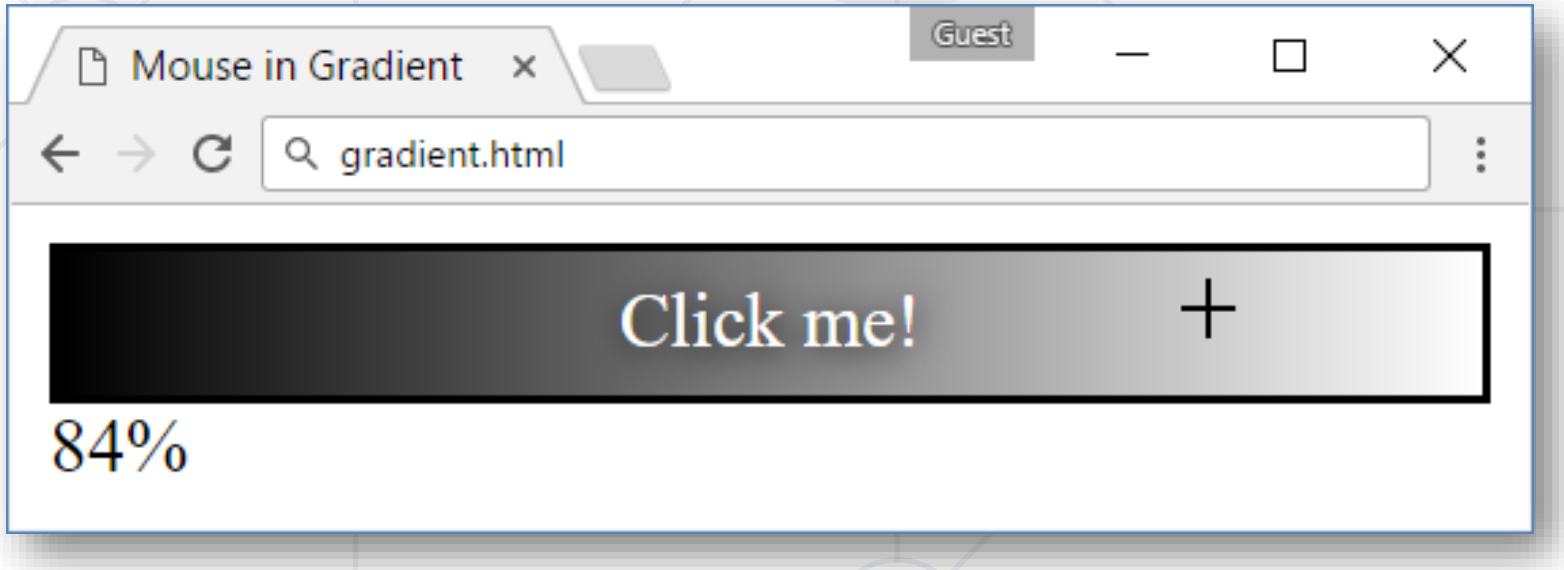
```
function solve() {
  let newElement = document.getElementById("newText").value;
  let list = document.getElementById("items");

  if (newElement.length === 0) return;

  let listItem = document.createElement("li");
  listItem.textContent = newElement;

  let remove = document.createElement("a");
  let linkText = document.createTextNode("[Delete]");
  // Continued on the next slide ...
```

Solution: Add / Delete Items

```
    remove.appendChild(linkText);
    remove.href = "#";
    remove.addEventListener("click", deleteItem);

    listItem.appendChild(remove);
    list.appendChild(listItem);

    function deleteItem() {
      listItem.remove();
    }
}
```

Problem: Mouse in Gradient

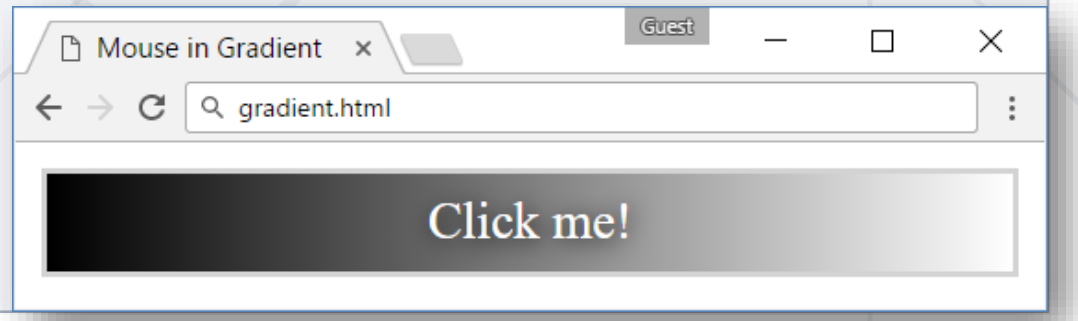
- A HTML page holds **linear gradient** box
  - Moving the mouse should show **percentage** [0% … 100%], depending on the **location of mouse**
  - Left side → **0%**; middle → **50%**; right side → **100%**



Check your solution here: *https://judge.softuni.org/Contests/Practice/Index/4367#3*

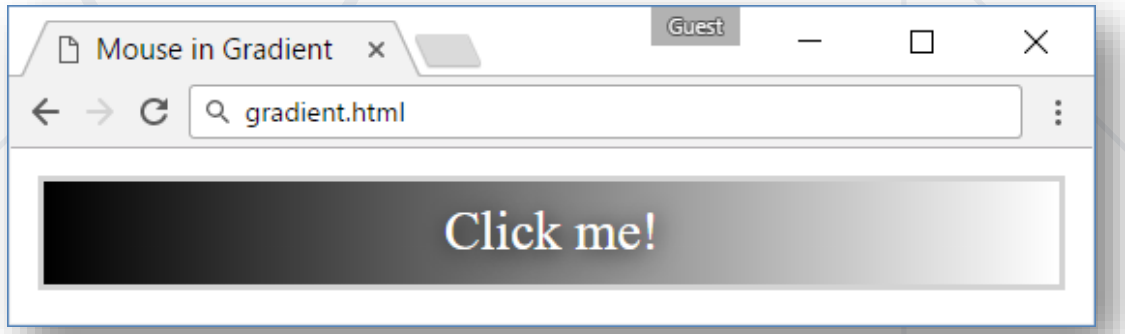
Problem: Mouse in Gradient – HTML

```
<html>
<head>
  <title>Mouse in Gradient</title>
  <link rel="stylesheet" href="gradient.css" />
  <script src="gradient.js"></script>
</head>
<body onload="attachGradientEvents()">
  <div id="gradient-box">
    <div id="gradient">Click me!</div>
  </div>
  <div id="result"></div>
</body>
</html>
```



Problem: Mouse in Gradient – CSS

```
#gradient-box {
  width: 300px;
  border: 2px solid lightgrey;
}
#gradient-box:hover {
  border: 2px solid black;
}
#gradient {
  height: 30px;
  color: white;
  text-shadow:
    1px 1px 10px black;
  text-align: center;
  line-height: 30px;
  background:
    linear-gradient(
    to right, black, white);
  cursor: crosshair;
}
```



Solution: Mouse in Gradient

Software University

```
function attachGradientEvents() {
  let gradient = document.getElementById('gradient');
  gradient.addEventListener('mousemove', gradientMove);
  gradient.addEventListener('mouseout', gradientOut);

  function gradientMove(event) {
    // Calculate the relative position of the mouse cursor within the 'gradient' element
    // by dividing the horizontal offset of the cursor by the total width
    let power = event.offsetX / (event.target.clientWidth - 1);
    // Convert the relative position to a percentage by multiplying it with 100
    power = Math.trunc(power * 100);
    document.getElementById('result').textContent = power + "%";
  }

  function gradientOut(event) {
    document.getElementById('result').textContent = "";
  }
};
```

Events Handler Execution Context

- In event handlers, **this** refers to the event **source element**

```
element.addEventListener("click", function(e) {
  console.log(this === e.currentTarget); // Always true
});
```

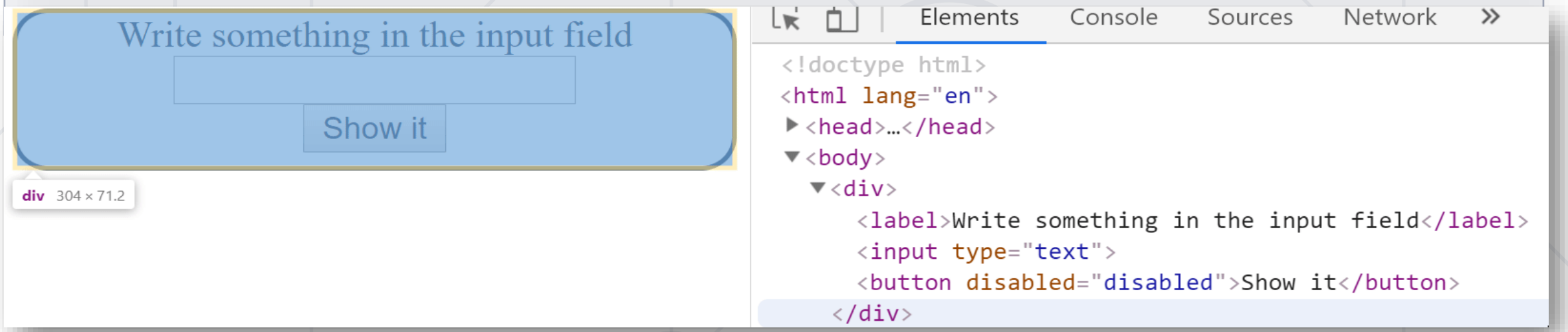
- Pay attention when using **object methods** as event listeners!
  - **this** may not behave as you expect with objects

Attaching Hover Handler

```
const button = document.getElementsByTagName('div')[0];
button.addEventListener('mouseover', function (e) {
    const style = e.currentTarget;
    const { backgroundColor } = style;

    if(backgroundColor === 'white'){
        targetStyles.backgroundColor = '#234465';
        targetStyles.color = 'white';
    } else {
        targetStyles.backgroundColor = 'white';
        targetStyles.color = '#234465';
    }}
);
```

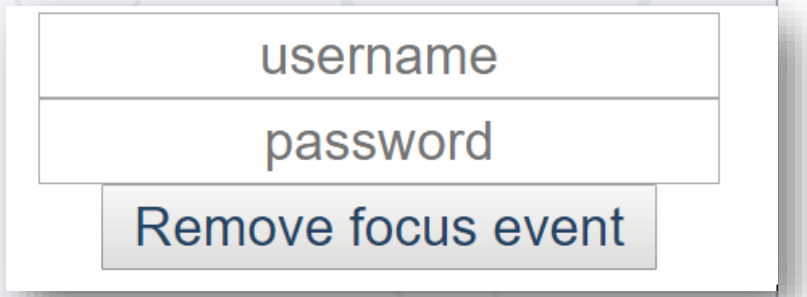
Attaching Input Handler

```
const inputField = document.getElementsByTagName('input')[0];
const button = document.getElementsByTagName('button')[0];

inputField.addEventListener('input', function () {
    button.setAttribute('disabled', 'false')
});
```



Remove Listeners

```
const password = document.querySelector('input[type="password"]');
const button = document.querySelector('button');
password.addEventListener('focus', focusEvent);

function focusEvent (){
    event.target.style.background = '#234465';
}

password.addEventListener('blur', (event) => {
    event.target.style.background = '';
});

button.addEventListener('click', () => {
    password.removeEventListener('focus', focusEvent);
});
```



Multiple Listeners

- The `addEventListener()` method also allows you to add many listeners to the same element, without overwriting existing ones:

```
element.addEventListener("click", myFirstFunction);
element.addEventListener("click", mySecondFunction);
element.addEventListener("mouseover", myThirdFunction);
element.addEventListener("mouseout", myFourthFunction);
```

*Note that you don't use the "**on**" prefix for the event use "**click**" instead of "**onclick**"*

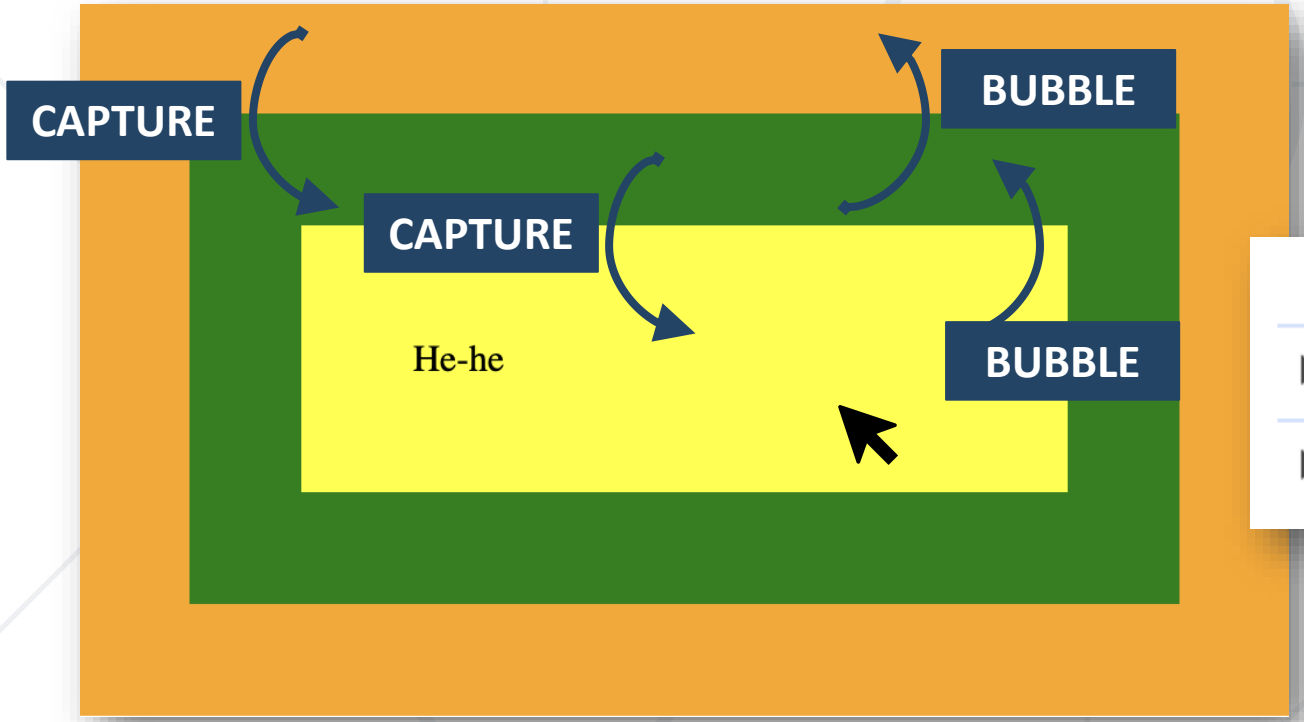
Event Propagation

```
<div class="orange">
   <div class="green">
      <div class="yellow">He-he</div>
   </div>
</div>
```

```
.orange { background: orange; }
.green { background: green; }
.yellow { background: yellow; }
```

```
const divElements = document.querySelectorAll("div");

function logText(e) {
   console.log(this, "clicked"); // currentTarget element
}

divElements.forEach((el) => el.addEventListener("click", logText));
```

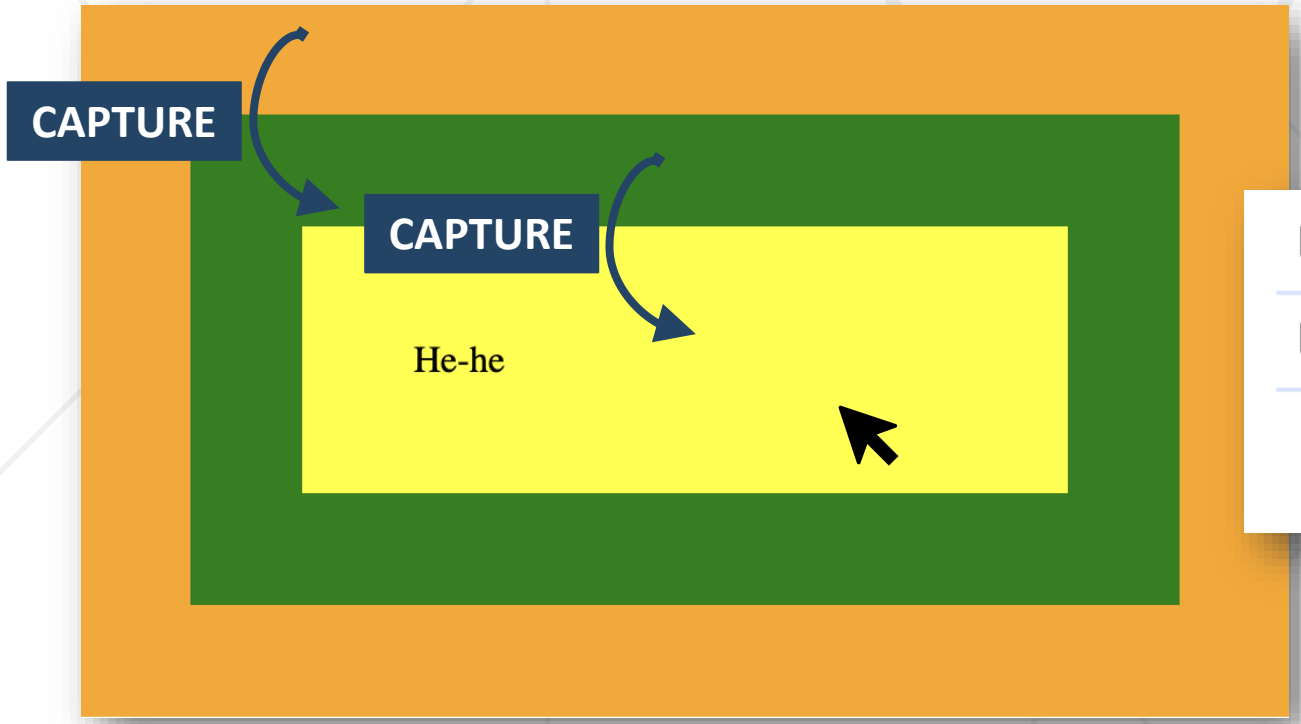
Event Propagation




```
<div class="yellow">He-he</div> 'clicked'
▶<div class="green">…</div> 'clicked'
▶<div class="orange">…</div> 'clicked'
```

Event Propagation

```
divElements.forEach((el) => el.addEventListener("click", logText, {
    capture: true // default is false
}));
```

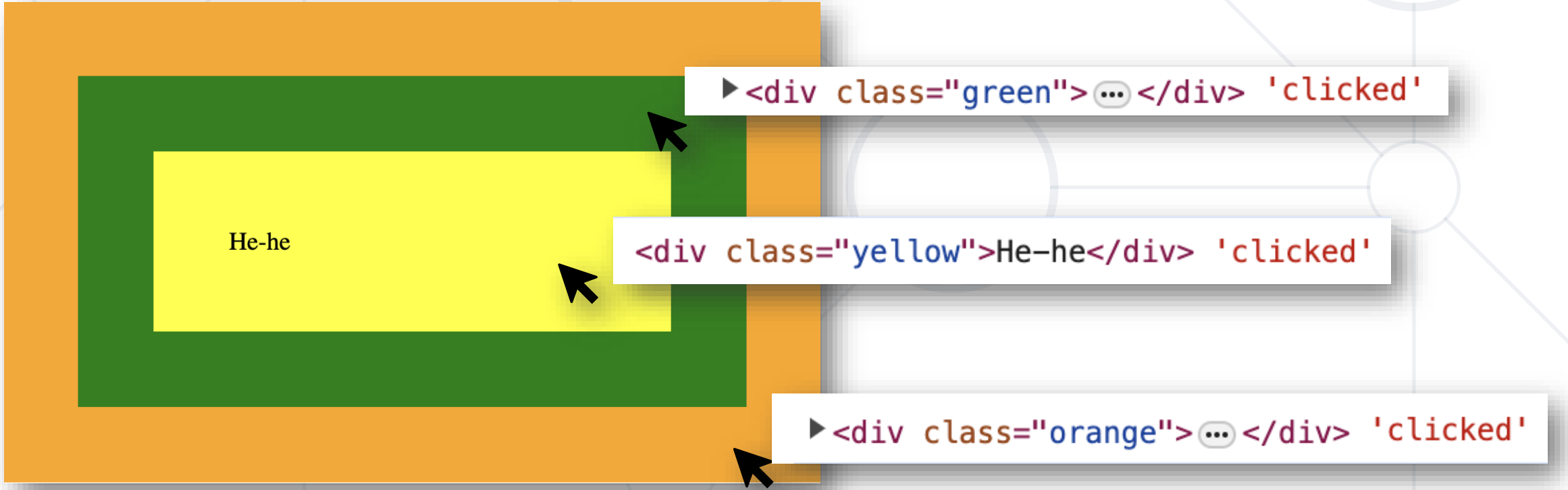



```
▶<div class="orange">…</div> 'clicked'
▶<div class="green">…</div> 'clicked'
  <div class="yellow">He-he</div> 'clicked'
```

Event Propagation

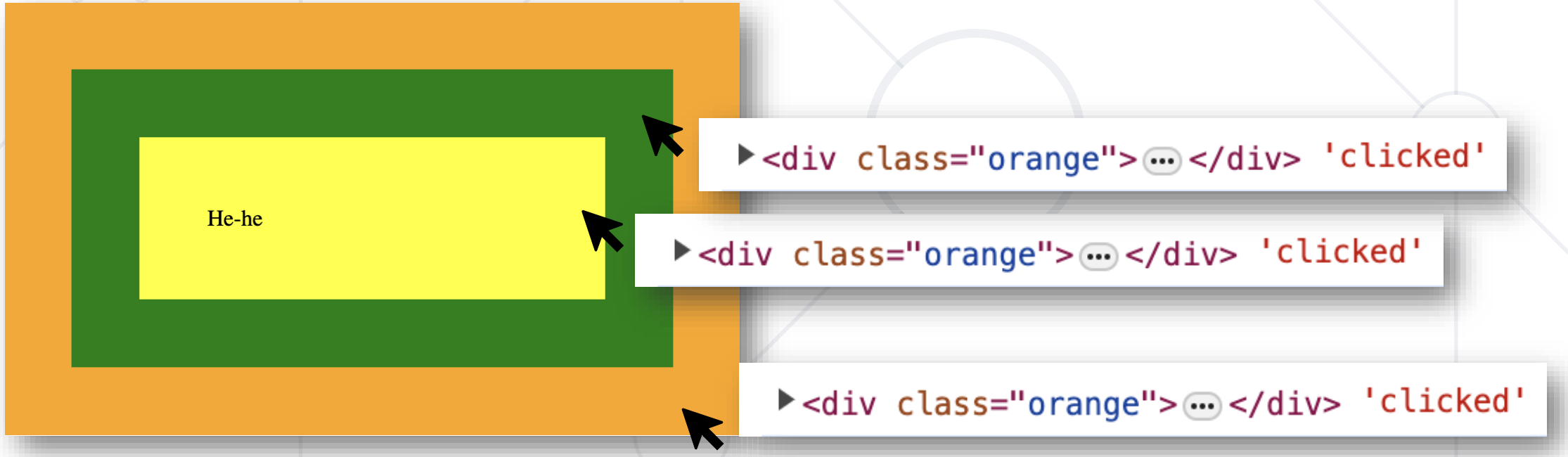
```
divElements.forEach((el) => el.addEventListener("click", logText));
```

```
function logText(e) {
    e.stopPropagation(); // prevents further propagation of the current event
        console.log(this, "clicked");
}
```

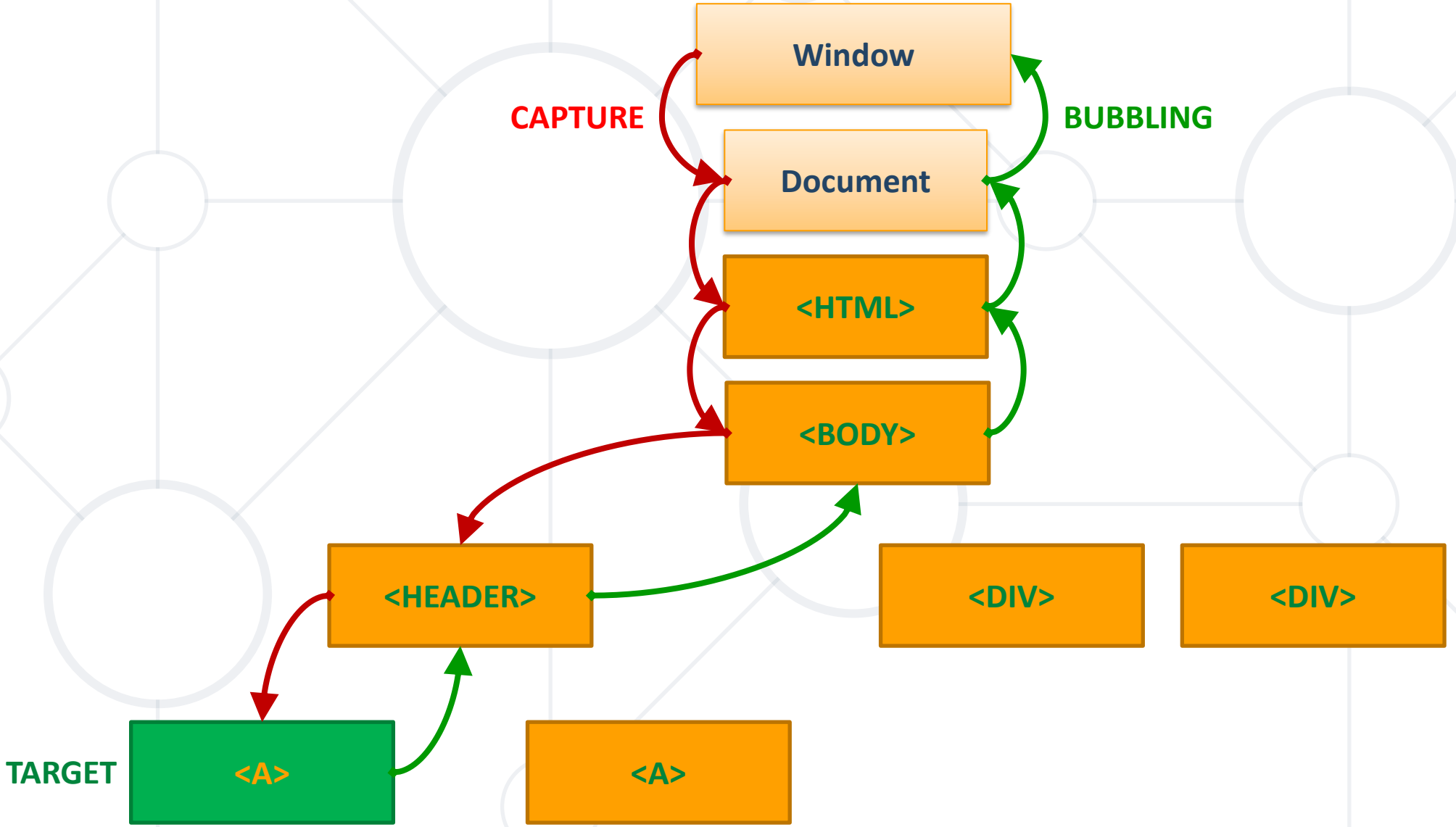


Event Propagation

```
divElements.forEach((el) => el.addEventListener("click", logText, {
    capture: true
}));
function logText(e) {
    e.stopPropagation();
    console.log(this, "clicked");
}
```



Event Propagation



Default Browser Behavior

- `preventDefault` stop the browser from executing default behavior, for example:
  - **Navigating** to a new page when <a> is clicked
  - Submitting **HTTP requests** via forms

```
<form id="myForm">
   <input type="text" placeholder="Username">
   <input type="password" placeholder="Password">
   <button type="submit">Submit</button>
</form>
```

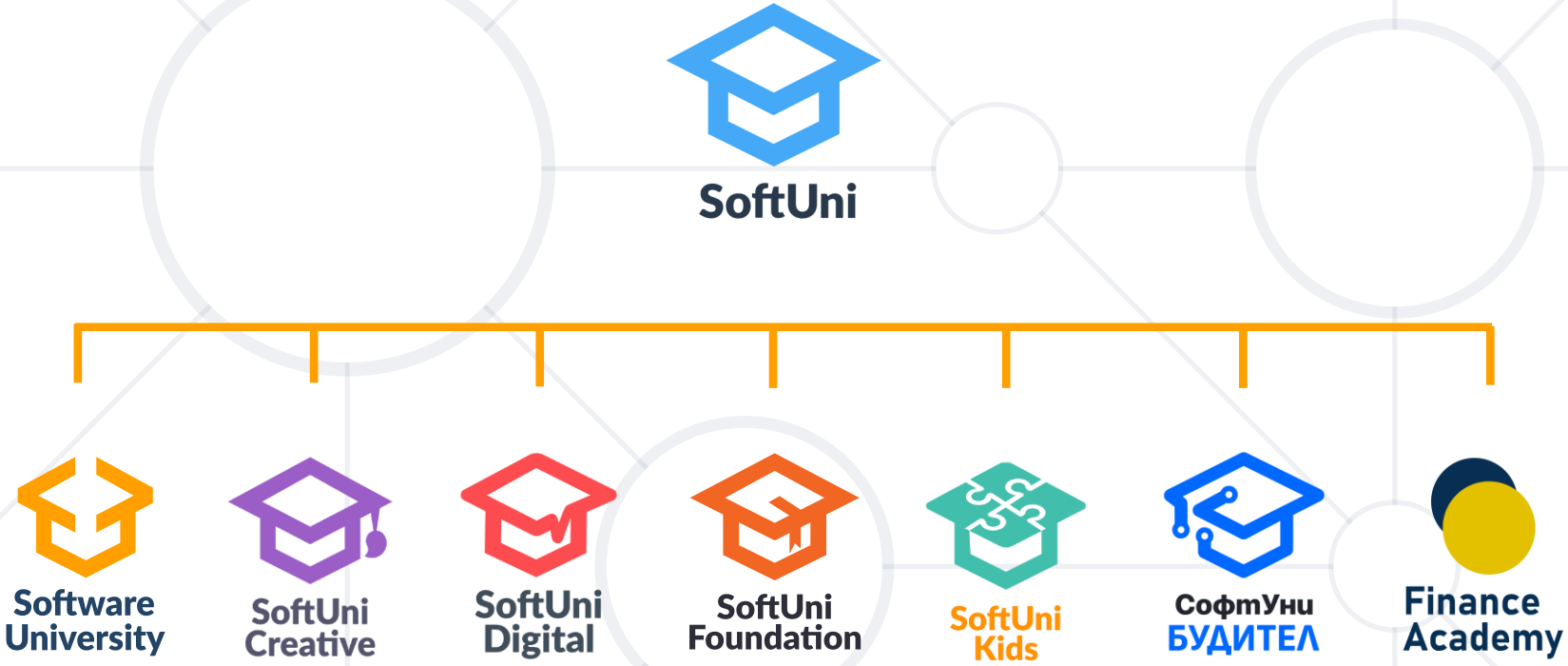
Default Browser Behavior

```
<script>
    const form = document.getElementById('myForm');

    form.addEventListener('submit', function (event) {
     // Prevent the default form submission behavior
     event.preventDefault();

    // We will add custom form validation logic here...
</script>
```

Summary

- The DOM tree can be manipulated by:
  - Creating and deleting elements
  - Moving elements between nodes
- User interaction triggers events
  - They can be listened to and handled
  - The handler receives event details
  - Events propagate through the DOM tree

Questions?



© SoftUni – https://about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg