Web Programming
JavaScript Part III.

Leander Jehl | University of Stavanger

Outline

- Today
  - More Event-driven programming
  - More Manipulating the DOM
  - Some tricks
  - OOP

Recap: assigning event handlers

A. In HTML

```
<div id=“green_div” onclick="handle();"></div>
```

B. In JavaScrips: using **on…**

```
let greenDiv = document.getElementById(“green_div");
greenDiv.onclick = handle;
```

C. In JavaScript: using **addEventListener**

```
let greenDiv = document.getElementById(“green_div");
greenDiv.addEventListener(“click”, handle);
```

Recap: event handlers with argument

To place string inside string, use single quotes.

A. In HTML

```
<div id="green_div" onclick="handle('blue');"></div>
```

B. In JavaScrips: using **on…**

```
let greenDiv = document.getElementById("green_div");
greenDiv.onclick = function(){ handle("hello");}
```

C. In JavaScript: using **addEventListener**

```
let greenDiv = document.getElementById("green_div");
greenDiv.addEventListener("click", function(){ handle("hello");});
```

Recap: event handlers with this

A. In HTML

```
function handle(element){ element.style.color = “blue” }
```

```
<div id=“green_div” onclick=“handle(this);”></div>
```

B. In JavaScrips: using **on…** (and **C**)

```
let greenDiv = document.getElementById(“green_div");
greenDiv.onclick = handle;
```

```
function handle(){ this.style.color = “blue” }
```

Recap: this and argument

A. In HTML

```
function handle(element, color){ element.style.color = color}
```

```
<div id="green_div" onclick="handle(this, 'blue');"></div>
```

B. In JavaScrips: using **on…** (and **C**)

```
let greenDiv = document.getElementById("green_div");
greenDiv.onclick = function(){ handle(this, "blue") }
```

Recap: run JS init function

A. In HTML no init function needed

```
<div id=“green_div” onclick="handle();"></div>
```

B. In JavaScrips, use **`window.onload`**

```
function init(){
    let greenDiv = document.getElementById(“green_div");
    greenDiv.onclick = handle;
}
window.onload = init;
```

Complete script will be executed once html is loaded.

C. In JavaScript, external

```
<script src=“myfile.js" defer></script>
```

D. Place at the end of **<body>** (not recommended)

Recap: changing element content

A. Using `innerHTML`

```
let greenDiv = document.getElementById(“green_div");
greenDiv.innerHTML = “<p>Some text</p>“;
```

B. Using `innerText` (for text only)

```
let greenDiv = document.getElementById(“green_div");
greenDiv.innerText = “text only“;
```

C. Using `value` on form elements

```
let nameInput = document.getElementById(“name-input”);
nameInput.value = “John Doe“;
```

Recap: changing attributes

- Set attribute value

```
let myimg = document.getElementById(“myimg");
myimg.src = “images/new_image.png“;
```

- Set style

```
let greenDiv = document.getElementById(“green_div");
greenDiv.style.backgroundcolor = “green“;
```

- Add/remove class

```
if (!greenDiv.classList.contains("border")) {
    greenDiv.classList.add("border");
}
else {
    greenDiv.classList.remove("border");
}
```

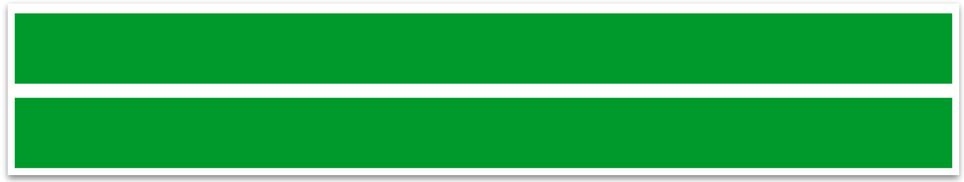
Exercises #1

github.com/dat310-spring20/course-info/tree/master/
exercises/js/more

Hint for Exercise #6

- Change the **`style.display`** or **`style.visibility`** property
- Remember the difference

CSS
```
#mydiv {
    style.display: none;
}
```



CSS
```
#mydiv {
    visibility: hidden;
}
```

DOM nodes

- Everything is a node
  - The document itself is a document node
  - All HTML elements are element nodes
  - All HTML attributes are attribute nodes
  - Text inside HTML elements are text nodes
  - Comments are comment nodes
- The **`nodeType`** property returns the type of the node

Traversing the DOM

- Finding child elements (excl. text and comment nodes)
  - **`childElementCount`** — number of child element an element has
  - **`children`** — child nodes of an element
  - **`hasChildNodes()`** — if an element has any child nodes
- Finding child elements (incl. text and comment nodes)
  - **`childNode`** — child nodes of an element
    - The number of elements can be accessed using **`childNode.length`**
- Finding parent element
  - **`parentNode`** — reference to the parent of the element

Example

examples/js/more/dom_traverse.html

```
function traverse(element, level) {
    let line = "";
    // indentation
    for (let i = 0; i < level; i++) {
        line += "  ";
    }
    // print element
    line += element.nodeName;
    console.log(line);

    // recursively traverse child elements
    if (element.hasChildNodes()) {
        for (let i = 0; i < element.children.length; i++) {
            traverse(element.children[i], level + 1);
        }
    }
}

window.onload = function () {
    traverse(document.body, 0);
}
```

Traversing the DOM (2)

- Some convenience properties
  - **`firstChild`** — first child node of an element
  - **`firstElementChild`** — first child element of an element
  - **`lastChild`** — last child node of an element
  - **`lastElementChild`** — last child element of an element
  - **`nextSibling`** — next node at the same node tree level
  - **`nextElementSibling`** — next element at the same node tree level
  - **`previousSibling`** — previous node at the same node tree level
  - **`previousElementSibling`** — previous element at the same node tree level
  - **`parentElement`** — parent element node of an element

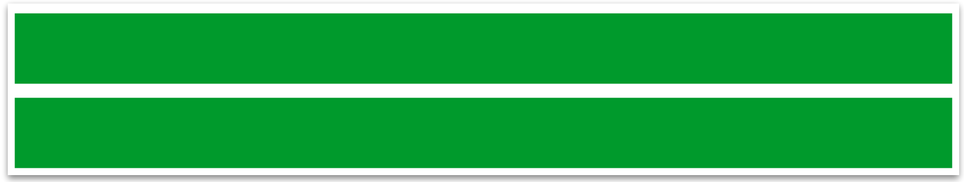
Exercises #2, #2b

github.com/dat310-spring20/course-info/tree/master/
exercises/js/more

Hint for Exercise #2

- Change the **`style.display`** or **`style.visibility`** property
- Remember the difference

CSS
```
#mydiv {
    style.display: none;
}
```



CSS
```
#mydiv {
    visibility: hidden;
}
```

Creating HTML elements

- To add a new HTML element
  - Create the element

```
let h2 = document.createElement(“h2");
```

  - Set the content of the element

```
h2.innerHTML = "Article header";
```

    - or

```
let text = document.createTextNode("Article header");
h2.appendChild(text);
```

  - Append it to an existing element (otherwise it won’t appear on the page)

```
let art1 = document.getElementById("article1");
art1.appendChild(h2);
```

Inserting new HTML element

- **`appendChild()`** adds new element after the last child element of the parent
- **`insertBefore()`** inserts before a specified child node

```
let newItem = document.createElement("li");
newItem.innerHTML = "Water";
// get the parent element
let list = document.getElementById("mylist");
// insert before the first child
list.insertBefore(newItem, list.children[0]);
```

Removing or replacing HTML elements

- To remove or replace a HTML element
  - You must know the parent of the element
    - If you identified the element, you can use the **`parentNode`** property to find its parent
- **`removeChild()`** — removes a given child element

```
let art1 = document.getElementById("article1");
art1.parentNode.removeChild(art1);
```

- **`replaceChild()`** — replaces a given child element

```
let art1 = document.getElementById("article1");
let art2 = document.createElement("article");
art1.parentNode.replaceChild(art2, art1);
```

Example

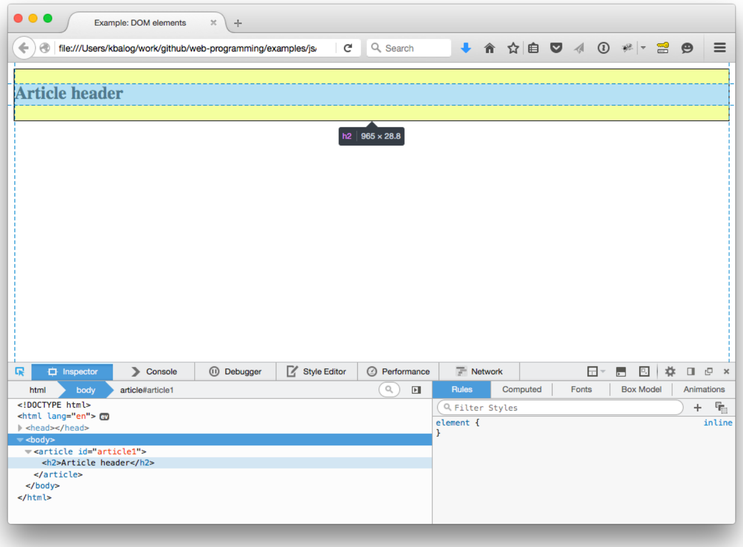
examples/js/more/dom_elements.html

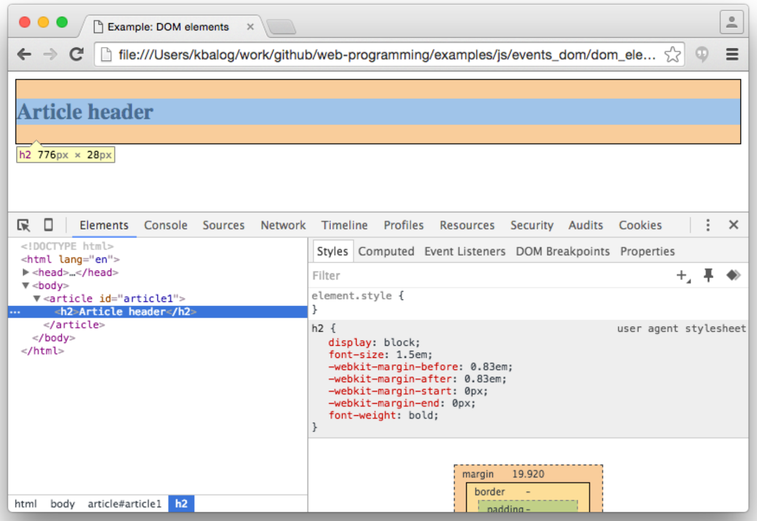
```
<script>
    function addArticleHeader() {
        // create a new heading
        let h2 = document.createElement("h2");
        // set the content of the new element
        h2.innerHTML = "Article header";
        // identify parent element
        let art1 = document.getElementById("article1");
        // append to parent element
        art1.appendChild(h2);
    }
</script>
```

```
<body>
    <article id="article1"></article>
</body>
```

Dev hint

- When using JS to change the DOM, use the browser's web inspector tool to see the modified HTML source
  - Viewing the page source will only show the initial HTML





Exercise #3, (#3b)

github.com/dat310-spring20/course-info/tree/master/
exercises/js/more

JS tricks

- Array iteration
- Query selectors
- Arrow functions

Array iteration

- Use **`for`** with index

```
for (let i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

- Use **`for of`**

```
for (let el of arr) {
    console.log(el);
}
```

- Use **`forEach(function… )`**

```
arr.forEach(function(element, index){
    console.log(element);
});
```

Query selectors

- Previous methods use id, tagName or class:

```
// get one element using its id
let element = document.getElementById(“mydiv”);

// get all elements with given tag
let array = document.getElementsByTagName(“div”);

// get all elements with given class
let array = document.getElementsByClassName(“nolog”);
```

querySelector

- **document.querySelector(“css-selector”)**
  - find first element that matches a css selector

```
// get one element using its id
let element = document.querySelector(“#mydiv”);

// get first element with given tag
let element = document.querySelector(“div”);

// get first element with given class
let element = document.querySelector(“.nolog”);

// get first matched by complex selector
let element = document.querySelector(“#mydiv .nolog+div”);
```

querySelectorAll

- **`document.querySelectorAll(“css-selector”)`**
  - find **all** elements that matches a css selector

```
// get all divs
let array = document.querySelectorAll(“div”);

// get all divs with class nolog
let array = document.querySelectorAll(“div.nolog”);

// get all divs without class nolog
let array = document.querySelectorAll(“div:not(.nolog)”);
```

querySelector/querySelectorAll

- `element.querySelectorAll(“css-selector”)`
  - find **all** elements that matches a css selector among descendants of an element

```
// get all divs inside this
let array = this.querySelectorAll(“div”);
```

Arrow functions

- Shorter anonymous functions

```
// get all elements with given tag
let divs = document.getElementsByTagName(“div”);
divs.forEach(function(elemen){
    console.log(element);
});

// using arrow function
divs.forEach((elemen)=>{ console.log(element) });
```

Arrow functions

- But they do not have own **this**

```
let mydiv = document.getElementById(“mydiv”);
mydiv.onclick = function(){
    // this refers to the mydiv
    this.style.backgroundColor = “blue”;
}

mydiv.onclick = ()=>{
    // this refers to the global object
    this.style.backgroundColor = “blue”;
}
```

Avoid using **this** inside arrow function.

https://www.w3schools.com/js/js_arrow_function.asp

Example

examples/js/more/event_listener.html

```
<script>
    function init() {
    document.querySelectorAll(“div”).forEach(
        (element)=> {element.addEventListener("click", showAlert)}
    );
    document.querySelectorAll(“div:not(.nolog)”).forEach(
        (element)=> {element.addEventListener("click", log)}
    );
    window.onload = init;
</script>
```

```
<body>
    <div></div>
    <div class=“nolog”></div>
    <div></div>
    <div class=“nolog"></div>
    <div></div>
</body>
```

setTimeout()

- Use **`setTimeout`** to schedule the execute a function (once)

```
function hello(){
    console.log(“hello”);
}

// say hello after one second
setTimeout(hello, 1000);
```

- Use **`setInterval`** to schedule execution regularly

```
// say hello every one second. Save the timer.
let timer = setInterval(hello, 1000);

... // do other stuff

// stop saying hello
clearInterval(timer);
```

Always clear your timers!

setTimeout with argument

- Pass function argument to **`setTimeout`** or **`setInterval`**

```
function makeBlue(element){
    element.style.backgroundColor = “blue”;
}

let mydiv = document.getElementById(“mydiv”);
setTimeout(makeBlue, 1000, mydiv);
```

setTimeout with this

- Function passed to **`setTimeout`** is executed with **`this`** set to global scope!

```
function Dog(name){
    this.name = name;
    this.info = function(){ console.log(this.name); }
    this.bark = ()=>{ console.log(this.name); }
}

let mydog = new Dog("Tiffy");

// this does not work
setTimeout(mydog.info, 1000);

// this does work since arrow function has no own scope
setTimeout(mydog.bark, 1000);
```

Avoid **this** or use arrow function!

- Read this stackoverflow answer

Exercises #4

github.com/dat310-spring20/course-info/tree/master/
exercises/js/more

Exercises #5-7

github.com/dat310-spring20/course-info/tree/master/
exercises/js/more

References

- W3C JavaScript and HTML DOM reference
  http://www.w3schools.com/jsref/default.asp
- W3C JS School
  http://www.w3schools.com/js/default.asp
- Mozilla JavaScript reference
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference