Algorithms and Programming IV
Web Application Development: CSS and JS (23-2)

Summer Term 2021 | 02.07.2021
Barry Linnert (Michael Tebbe)
Cascading Style Sheets (CSS)

Style sheets can be used to specify how tables should be rendered, how lists should be presented, what colors should be used on the webpage, what fonts should be used and how big/small they are, etc.

Cascading Style Sheets, can be defined at three different levels:

1. Inline style sheets apply to the content of a single HTML element (style-Attribute)
2. Document style sheets <style> … </style>
3. External style sheets can be linked and applied to numerous documents:

```html
<head>
  ...
  <link rel="stylesheet" href="css/styles.css?v=1.0">...
</head>
```
CSS Syntax
CSS - Selectors

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them, e.g. all children)
- Pseudo-class selectors (select elements based on a certain state, e.g. visited/unvisited link)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

```css
.center {
  text-align: center;
}

div p {
  color: yellow;
}

a:link {
  color: #FF0000;
}
a:visited {
  color: #00FF00;
}
p::first-line {
  color: #ff0000;
}
[title~="flower"] {
  border: 5px solid yellow;
}
```
CSS - ways to style an html element

All parts of an HTML-Element can be styled.

Examples:

**Text**

```css
h1 {
    color: blue;
    font-family: Verdana;
    font-size: 12px;
}
```

**Lists**

```css
ul {
    list-style-type: circle;
}
```
CSS - Box Model

- Margin
- Border
- Padding
- Content
CSS - Specificity

If there are two or more conflicting CSS rules that point to the same element, the rule that is applied is determined based on specificity.

```
<h1 id="example">Example Headline</h1>
```

```css
h1 {
  color: red;
}
```

```css
#example {
  color: blue;
}
```

Element-Selector = low Specificity vs. ID-Selectors = high Specificity

=> Style in ID selector wins and text will be displayed in blue
CSS - Frameworks

Materialize, Foundation, Bulma…. Example: Bootstrap

- open source front-end toolkit
- responsive grid system
- prebuilt components
- JavaScript plugins

```html
<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="css/bootstrap.css">
```
Web Application Development I

JavaScript (JS)
What is JavaScript?

JavaScript is an implementation of the ECMA Script language standard. ECMA-262 is the official JavaScript standard. (http://www.ecma-international.org/publications/standards/Ecma-262.htm)

JavaScript was invented by *Brendan Eich* at Netscape (with Navigator 2.0), and has appeared in all browsers since 1996.

The official standardization was adopted by the ECMA organization (an industry standardization association) in 1997.

The ECMA standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.
What can JavaScript do?

JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages.

JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element.

JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element.
What can JavaScript do?

JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing.

JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser.

JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer.
Lexical Structure

- JavaScript programs are written using the Unicode character set. It is a case-sensitive language.
- In JavaScript, you can usually omit the semicolon between two statements if those statements are written on separate lines.
- JavaScript does not treat every line break as a semicolon: it usually treats line breaks as semicolons only if it can’t parse the code without the semicolons.

Examples

```
var a = 3;
console.log(a);
```

```
var y = x + f(a+b).toString();
```

```
return; true;
```

```
x; ++y;
```

Interpretation

```
var a; a = 3;
console.log(a);
```

```
var y = x + f(a+b).toString();
```

```
return; true;
```

```
x; ++y;
```
JavaScript Object Hierarchy
JavaScript Basics

Declaration
- Explicit: `var i = 12; // no 'var' in declaration`
- Implicit: `i = 12;`

Variable scope
- Global: Declared outside functions and any variable that is implicitly defined
- Local: Explicit declarations inside functions

Control and looping
- ‘if’, ‘switch’ statement
- `while`, and `do-while` loops (break and continue keywords)

Tutorial at
http://www.w3schools.com/js/
Not object-oriented but object-based

```javascript
function Student(studentName, studentAge) {
    this.name = studentName;
    this.age = studentAge;
}

var someStudent = new Student("Michael", 21);
```

Tutorial at [http://www.w3schools.com/js/](http://www.w3schools.com/js/)
The document objects allow printing directly into the browser page (amongst other things).

The `window` object is implied.

Example: Writing in text or HTML with script or without line-break

```javascript
    document.write("I am <B>BOLD</B>");
    document.writeln("I am <U>underlined</U>");
```

What is the `window` object?

- It is part of the Browser Object Model (BOM)
- BOM is a collection of objects that represent the browser and the computer screen
- Objects are accessible through the global objects `window` and `window.screen`
- The `window` object is global. It represents the host environment of all other JS objects
Browser Window Object

`window` object is a JavaScript representation of a browser window

- **closed** - A boolean value that indicates whether the window is closed
- **defaultStatus** - This is the default message that is loaded into the status bar when the window loads

**Example:**
```
window.defaultStatus = "This is the status bar";
```

Selected *built in functions*

- `alert("message")` - string passed to the alert function is displayed in an alert dialog box.
- `window.close()` - function will close the current window or the named window.
- `confirm("message")` - string passed to the confirm function is displayed in the confirm dialog box.
- `focus()` - function will give the focus to the window.
- `open("URLname","Windowname",["options"])` - new window is opened with the name specified by the second parameter.
Form Object

The Form object is a property of the document object. This corresponds to an HTML input form constructed with the FORM tag. A form can be submitted by calling the JavaScript submit method or clicking the form submit button.

Form objects can be accessed by
window.document.myForm OR window.document.forms[0]
Form Object – Properties

Selected properties

**action** - This specifies the URL and CGI script file name the form is to be submitted to. It allows reading or changing the ACTION attribute of the HTML FORM tag.

**target** - The name of the frame or window the form submission response is sent to by the server. Corresponds to the FORM TARGET attribute.

**length** - The number of fields in the elements array, i.e. the length of the elements array.

**method** - This is a read or write string. It has the value "GET" or "POST".
Form Object – objects and methods

Form Objects (one example)
- **text** - A GUI text field object. Methods are `blur()`, `focus()`, and `select()`
- Attributes
  - `defaultValue` - The text default value of the text field
  - `name` - The name of the text field
  - `type` - Type is "text"
  - `value` - The text that is entered and appears in the text field. It is sent to the server when the form is submitted
- Example “accessing form field values”: `window.document.myForm.firstname.value`
Form Object – objects and methods

Form object methods
• reset() - Used to reset the form elements to their default values
• Example `window.document.myForm.reset();`
• submit() - Submits the form as though the submit button were pressed by the user
• Example `window.document.myForm.submit();`
Event handlers

Events are actions that occur usually as a result of something the user does. For example, clicking a button is an event, as is changing a text field or moving the mouse over a hyperlink. Other events are `click`, `change`, `focus`, `load`, `mouseover`, `mouseout`, `reset`, `submit`, `select` ...

You can use event handlers, such as `onChange` and `onClick`, to make your script react to events.

Examples

```html
<input type="button" onClick="javascript:doButton()">
<select onChange="javascript:doChange()">
<a onClick="javascript:doSomething()"> </a>
<form onSubmit="javascript:validate()">
```
Frontend Overview: Example

Content

```html
<head>
  <script src="myJS.js">
  <link rel="stylesheet" type="text/css" href="myCSS.css">
  ...
</script>
```

Behavior

```javascript
script.js
```

Presentation

```css
style.css
```
From HTML to DOM

Browsers must render HTML documents (i.e., apply CSS and execute JavaScript)

1. GET HTML from server and receive as text/html document
2. Parse document and deal with any errors by “fixing them”
3. Interpret document as if it had been error-free
4. GET all additional resources (CSS, images, JavaScript, …)
5. Build internal model (DOM) based on error-free interpretation
6. Apply CSS rules to determine styling of document (e.g., margins and font sizes)
7. Render into visual structure
8. Start executing JavaScript code
9. Listen for events (keyboard, mouse, timer) and execute code
W3C Document Object Model (DOM)

It describes a tree structure of all HTML elements, including attributes and the text they contain. (http://www.w3.org/DOM/DOMTR)

DOM is under constant revision

- DOM0 was invented by Netscape (backing LiveScript/JavaScript)
- DOM1 was the first DOM version produced by the W3C
- DOM2 is the currently available stable version of DOM
- DOM3 is highly modularized and still under development
DOM representation of an HTML page

Each HTML document loaded into a browser window becomes a Document object

• Provides access to all HTML elements in a page, from within a script.
• Is also part of the Window object, and can be accessed through the window.document property.

DOM describes how all elements in an HTML page are related to the topmost structure: the document itself.
Accessing Elements

By name

document.getElementsByTagName(“td”)[indexOfColumn]

By ID

document.getElementById(“mytable”)

Walk the DOM Tree