{POWER.CODERS}

Introduction to JavaScript

Agenda

Today we will learn about

- What JavaScript is
- Basic syntax
- First data types
- Variables

JavaScript on the web

Up to now you learned how to do so called static web sites. They are called static because the content can not be changed.

JavaScript on the web

Up to now you learned how to do so called static web sites. They are called static because the content can not be changed.

With JavaScript we make the web sites interactive.

Web use cases

Web use cases

- AJAX loaded content (loading parts of the content without refreshing the site)
- Include external content (e.g. add Twitter feed)
- Form validation and process data
- Overlay elements and lightboxes
- Sliders, tabs and accordions
- Website tracking
- Drawing and animation

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior
- Other browsers included their own versions of JavaScript

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior
- Other browsers included their own versions of JavaScript
- The result was that there was **no standard**

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior
- Other browsers included their own versions of JavaScript
- The result was that there was **no standard**
- Web developers had a hard time to make sure their web sites were **browser compatible**

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior
- Other browsers included their own versions of JavaScript
- The result was that there was **no standard**
- Web developers had a hard time to make sure their web sites were **browser compatible**
- **ECMAScript** (ES) was created at first standard in 1997

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior
- Other browsers included their own versions of JavaScript
- The result was that there was **no standard**
- Web developers had a hard time to make sure their web sites were **browser compatible**
- **ECMAScript** (ES) was created at first standard in 1997
- The current version is ES2021 (12th edition), but the browsers lag behind. ES5 (2009) is the one all modern browsers understand.

SMALL HISTORY LESSON

- Netscape invented JavaScript 1995 to **add actions** to the web and make their browser superior
- Other browsers included their own versions of JavaScript
- The result was that there was **no standard**
- Web developers had a hard time to make sure their web sites were **browser compatible**
- **ECMAScript** (ES) was created at first standard in 1997
- The current version is ES2021 (12th edition), but the browsers lag behind. ES5 (2009) is the one all modern browsers understand.
- Browsers implement specific features of newer ES versions all the time. Use Caniuse to check for support.

Let's have a first try

Code along with me

We will do that in the online session on Friday.

Internal vs. External

Internal

```
<body>
...
  <script>
    window.alert("Hello World!");
  </script>
...
</body>
```

Internal vs. External

Internal

```
<body>
...
  <script>
    window.alert("Hello World!");
  </script>
...
</body>
```

`<script>` tags can be inserted in head and body.

Internal vs. External

Internal

```
<body>
...
  <script>
    window.alert("Hello World!");
  </script>
...
</body>
```

`<script>` tags can be inserted in head and body.

Each instruction in JS is a **statement**.

Internal vs. External

Internal

```
<body>
...
  <script>
    window.alert("Hello World!");
  </script>
...
</body>
```

`<script>` tags can be inserted in head and body.

Each instruction in JS is a **statement**.

Statements are separated by **semicolons**.

Internal vs. External

External

```
<body>
...
  <script src="myScript.js"></script>
...
</body>
```

Internal vs. External

External

```
<body>
...
  <script src="myScript.js"></script>
...
</body>
```

Normally you place the javascript code into a **external file** and load the file at the **end of the body.**

Internal vs. External

External

```
<body>
...
  <script src="myScript.js"></script>
...
</body>
```

Normally you place the javascript code into a **external file** and load the file at the **end of the body.**

- It's easier to maintain
- It's faster to load

What is Programming?

- Programming is the art of breaking a problem down into smaller problems.
- Finding the solutions to these smaller problems.
- Putting these pieces back together.

Building blocks of Programming:

- Data Types
- Variables
- Conditions
- Loops
- Functions

Building blocks of Programming:

- Data Types
- Variables
- Conditions
- Loops
- Functions

You have already used a function `alert("Hello World!);`

Data types

Indicate in JavaScript what types of value are possible and which kind of operations.

Data types

Indicate in JavaScript what types of value are possible and which kind of operations.

- Number
- String
- Boolean
- Undefined
- Null
- Symbol (new in ES6)
- Object

Data types

Indicate in JavaScript what types of value are possible and which kind of operations.

- Number `3.141502653589793`
- String
- Boolean
- Undefined
- Null
- Symbol (new in ES6)
- Object

Data types

Indicate in JavaScript what types of value are possible and which kind of operations.

- Number `3.141502653589793`
- String `"Hello world"`
- Boolean
- Undefined
- Null
- Symbol (new in ES6)
- Object

Data types

Indicate in JavaScript what types of value are possible and which kind of operations.

- Number `3.141502653589793`
- String `"Hello world"`
- Boolean `true`
- Undefined
- Null
- Symbol (new in ES6)
- Object

How to write strings

```
'This is a string' // single quotes are mostly used
"This is also a string" // double quotes can be used, but should be avoided

'What's up?' // This is an error
'What\s up?' // backslash to "escape" the quotes
"What's up?" //legit use of double quotes

"They said \"What's up?\"."   // more quote escaping
'They said "What\'s up?".'    // ... or use single quotes
```

TEMPLATE STRINGS

Easier to write strings with variables inside. Use back ticks (` ) and `${}` around your variables for template strings.

```
let name = "Susanne";
const greeting = "Hello " + name + ", great to see you again.";
const betterGreeting = `Hello ${name}, great to see you again.`;
```

CHANGE TO TEMPLATE STRINGS

```
const message = "Hello " + firstName + " have I met you before?
  I think we met in " + city + " last summer no???";
```

VARIABLES

In variables you store data

```
//This is a comment. It won't get read by your browser.

//The next line declares a variable.
var x = 10;
```

- The keyword **var** defines a variable.
- **x** is the name of the variable.
- **10** is the value of the variable.
- The **equal sign (=)** is called the assignment operator.

VARIABLES AND DATA TYPES

```
/* Multi-line comments are possible as well.
You can write any number you want. */
const pi = 3.141502653589793;

let name = 'Powercoders';
//This is called a string. It is in single or double quotes.

var bool = True;
//This is a Boolean.
```

Scope

In JavaScript each function creates a new scope.
Scope determines the accessibility (visibility) of the declared variables.

Variables defined inside a function are not accessible (visible) from outside the function.

GLOBAL SCOPE

You are in the **global scope**, also called **root scope** by default if you are using JavaScript, the **window object**.

GLOBAL VARIABLES

A **global variable** is short for a variable defined in global scope.

Variables declared **outside** of any functions are **global**.

A global variable can be used anywhere **after** its declaration.

GLOBAL VARIABLES

A **global variable** is short for a variable defined in global scope.

Variables declared **outside** of any functions are **global**.

A global variable can be used anywhere **after** its declaration.

Declare all global variables at the top of your JS file.

Local scope

If you declare a variable inside a code block (e.g. function), you create a **local scope**, also called **child scope** or **function scope**

If you have variables inside your local scope, which are not declared there, it will check the global scope for the variable.

Local scope

If you declare a variable inside a code block (e.g. function), you create a **local scope**, also called **child scope** or **function scope**

If you have variables inside your local scope, which are not declared there, it will check the global scope for the variable.

Scoping rules on variables were always very confusing and the reason for many bugs.

`let` vs. `var`

- `var` is function-scoped. Every variable declared inside the function is only accessible inside the function.
- `let` is block-scoped (block is anything surrounded by {}). Every variable declared inside a {} block is only accessible inside that block. `const` as well.

`let` vs. `var`

- `var` is function-scoped. Every variable declared inside the function is only accessible inside the function.
- `let` is block-scoped (block is anything surrounded by {}). Every variable declared inside a {} block is only accessible inside that block. `const` as well.

Best practice: use `let` over `var`. Use `const` for variables which do not change.

Naming of variables

JavaScript is **case sensitive**.

```
// These are two different variables
let name = 'Powercoders';
let Name = 'Powercoders';
```

Naming of variables

JavaScript is **case sensitive**.

```
// These are two different variables
let name = 'Powercoders';
let Name = 'Powercoders';
```

- The first character **must be** a letter, an underscore (_) or a dollar sign ($)

Naming of variables

JavaScript is **case sensitive**.

```
// These are two different variables
let name = 'Powercoders';
let Name = 'Powercoders';
```

- The first character **must be** a letter, an underscore (_) or a dollar sign ($)
- Numbers are **not allowed** as first character

Naming of variables

JavaScript is **case sensitive**.

```
// These are two different variables
let name = 'Powercoders';
let Name = 'Powercoders';
```

- The first character **must be** a letter, an underscore (_) or a dollar sign ($)
- Numbers are **not allowed** as first character
- Variable names cannot include **mathematical or logical operators**, for instance (*) or (+)

Naming of variables

JavaScript is **case sensitive**.

```
// These are two different variables
let name = 'Powercoders';
let Name = 'Powercoders';
```

- The first character **must be** a letter, an underscore (_) or a dollar sign ($)
- Numbers are **not allowed** as first character
- Variable names cannot include **mathematical or logical operators**, for instance (*) or (+)
- JavaScript variables **must not contain spaces**.

Naming of variables

JavaScript is **case sensitive**.

```
// These are two different variables
let name = 'Powercoders';
let Name = 'Powercoders';
```

- The first character **must be** a letter, an underscore (_) or a dollar sign ($)
- Numbers are **not allowed** as first character
- Variable names cannot include **mathematical or logical operators**, for instance (*) or (+)
- JavaScript variables **must not contain spaces**.
- Avoid reserved words: List on w3schools

VARIABLES

A variable can be declared without value.

```
let my_variable;
```

VARIABLES

A variable can be declared without value.

```
let my_variable;
```

A variable declared without value is **undefined**.

VARIABLES

A variable can be declared without value.

```
let my_variable;
```

A variable declared without value is **undefined**.

The data type `undefined` means the variable has not been assigned.

Use of the console

Dev Tools include a REPL (read, evaluate, print, loop), better known as console.

```
console.log('Hello ' + 'world');
```

Result: 'Hello world'

```
let x = 'This is a ';
x + 'string'; // add string to variable
```

Result: 'This is a string'

```
let y = 1;
x + y;
```

Result: 'This is a 1'

Console cheatsheet

- `clear()` : clears the console
- `alert()` : writes output in a popup
- `prompt()` : asks for input in a popup
- `Number()` : converts a string to a number
- The arrow key up shows you a history of your console entries
- Shift + Enter creates a new line without executing it

EXPRESSIONS

Operator	Operation	Example	Result
`+`	Addition	`2 + 2`	`4`
`-`	Subtraction	`2 - 2`	`0`
`/`	Division	`3 / 2`	`1.5`
`*`	Multiplication	`5 * 2`	`10`
`%`	Remainder / modulus	`9 % 2`	`1`
		`8 % 2`	`0`
`**`	Exponention (x^y)	`2**3`	`8`

Increment & Decrement

to add or substract 1 from a number.

Operator	Example	Result
`var++`	`var a=0, b=10;` `var a=b++;`	a=10 / b=11
`++var`	`var a=0, b=10;` `var a=++b;`	a=11 / b=11
`var--`	`var a=0, b=10;` `var a=b--;`	a=10 / b=9
`--var`	`var a=0, b=10;` `var a=--b;`	a=9 / b=9

Assignment operators

Operator	Example	is equivalent to
`=`	`x = y`	`x = y`
`+=`	`x += y`	`x = x + y`
`-=`	`x -= y`	`x = x - y`
`*=`	`x *= y`	`x = x * y`
`/=`	`x /= y`	`x = x / y`
`%=`	`x %= y`	`x = x % y`

Online ressources

- MDN JavaScript Grammar & Types
- MDN JavaScript basics
- JavaScript introduction
- JavaScript tutorial
- Using variables in JavaScript

Online exercises

- w3schools JS exercise
- Freecode camp JS exercise (first 28)
- JavaScript exercises on exercism.io (with account)

EXERCISES

Fortune teller (45 min)

Why pay a fortune teller when you can just program your fortune yourself?

- Store the following into variables: number of children, partner's name, geographic location, job title.
- Output your fortune to the screen / console like so: "You will be a X in Y, and married to Z with N kids."

Age calculator (45 min)

Want to find out how old you'll be? Calculate it!

- Store your birth year in a variable.
- Store a future year in a variable.
- Calculate your 2 possible ages for that year based on the stored values.
  For example, if you were born in 1988, then in 2026 you'll be either 37 or 38, depending on what month it is in 2026.
- Output them to the screen/console like so: "I will be either NN or NN in YYYY", substituting the values.

Say hello (30 min)

Create a program in JavaScript that prompts for your name and prints a greeting using your name.

Counting the number of characters (45 min)

Create a program in JavaScript that prompts for an input string and displays output that shows the input string and the number of characters the string contains.

Printing quotes (30 min)

Create a program that prompts for a quote and an author. Display the quotation and author as shown here:
[Author] says, "[Quote]" (Replace [Author] and [Quote] with the actual values