EGOS!®
the education company

JavaScript/TypeScript

Trainer: Georg Praßl BSc.

Agenda

EGOS!
the education company

- Funktionen
  - Grundlagen
  - Variablen
  - Funktionen
- Objekte
- Klassen
- DOM
- Debugging
- TypeScript
  - Basics
  - HowTo

Was ist JavaScript

EGOS!®
the education company

- Die bekannteste Programmiersprache
- JS ist die Programmiersprache des Internets
  - Jedes TypeScript File wird am Ende wieder in JS übersetzt
- JS ist leicht zu lernen ABER es ist auch leicht damit Fehler zu machen

Was kann JS

EGOS!® the education company

- HTML Content ändern → getElementById().innerHTML()
- Wert von HTML Attribute ändern → zb. getElementById().src bei einem <img/>
- HTML Styles ändern → getElementById().style.font
- HTML Elemente verschwinden und anzeigen lassen → getElementById().style.display = „none“

JS – Daten anzeigen

EGOS!® the education company

- innerHTML → Schreibt in HTML Element
- Documtent.write() → Schreibt in den HTML output
  - Wenn nach Document.load() aufgerufen wird wird gesamtes bereits Existierende HTML gelöscht!
- Window.alert → Schreibt in die Alarm Box

Wichtigsten JS - Keywords

EGOS!® the education company

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

JS-Variablen

EGOS!
the education company

- Const → konstante Referenz zu einem Wert, Wert nicht mehr änderbar, Blockscope, müssen bei Deklaration einen Wert bekommen
  - Bei einem Array, Object, Funtion oder RegEx immer const
  - IE10 oder früher nicht supported
- Let → Variablen, veränderbar aber nur einmal deklarierbar, BlockScope
- Var → Variablen die Böse sind! Don‘t use them!

JS-Scopes

EGOS!®
the education company

- Block Scope
  - Innerhalb eines Blocks { }
- Functional Scope
  - Innerhalb einer Funktion
- Global Scope
  - Wenn außerhalb von Funktion deklariert → von überall aus erreichbar und veränderbar

JS - Datentypen

EGOS!
the education company

- Number
- String
- Object
- Booleans
- Arrays

JS - Functions

EGOS!
the education company

- function name(param1, param2, param3, ...){ }
- Funktionen werden ausgeführt durch
  - Events
  - Aufrufe
  - Automatically (Selbst Aufruf)
- Funktionen können als Variablen zugewiesen werden oder aufgerufen durch ()
- Funktionen sind grundsätzlich Objekte
- Seit ES6: () => { } oder (x,y) => x*y
  - Nicht in IE11 oder früher supported

JS – Function Parameters

EGOS!® the education company

- Regeln
  - Keine Definition welche Datentypen die Parameter haben müssen
  - Funktionen machen kein Type-Checking bei den übergebenen Argumenten
  - Funktionen überprüfen nicht die Anzahl der Argumente welche übergeben werden
- Seit ES6 sind default Parameter möglich
- Arguments → ist ein Array mit allen übergebenen Argumenten
- Argumente werden als Wert übergeben
- Objekte werden als Referenz übergeben

JS - this

EGOS!®
the education company

In an object method, this refers to the **object**.
Alone, this refers to the **global object**.
In a function, this refers to the **global object**.
In a function, in strict mode, this is undefined.
In an event, this refers to the **element** that received the event.
Methods like call(), apply(), and bind() can refer this to **any object**.

JS – Function Call()

EGOS!® the education company

```
1. const person = {
     fullName: function(city, country) {
       return this.firstName + "
   " + this.lastName + "," + city + "," + country;
     }
   }

   const person1 = {
     firstName:"John",
     lastName: "Doe"
   }

   person.fullName.call(person1, "Oslo", "Norway");
```

JS – Function Apply()

EGOS!® the education company

```
const person = {
  fullName: function(city, country) {
    return this.firstName + "
" + this.lastName + "," + city + "," + country;
  }
}

const person1 = {
  firstName:"John",
  lastName: "Doe"
}

person.fullName.apply(person1, ["Oslo", "Norway"]);
```

JS – Function bind()

EGOS!® the education company

```
const person = {
  firstName:"John",
  lastName: "Doe",
  display: function () {
    let x = document.getElementById("demo");
    x.innerHTML = this.firstName + "
   " + this.lastName;
  }
}

let display = person.display.bind(person);
setTimeout(display, 3000);
```

JS – Function Closures 1

EGOS!® the education company

- Problem
-
```
// Initiate counter
let counter = 0;

// Function to increment counter
function add() {
  counter += 1;
}

// Call add() 3 times
add();
add();
add();
// The counter should now be 3
```

JS – Function Closures 2

EGOS!® the education company

- Lösung

```
const add = (function () {
  let counter = 0;
  return function () {counter
+= 1; return counter}
})();

add();
add();
add();
```

Objekte 1

EGOS!® the education company

- Fast alles in JS sind Objekte
  - Boolens, Numbers, String (wenn mit „new“ definiert)
  - Dates, Maths, RegEx, Arrays, Functions, Objects
- Alle Werte, ausgenommen primitive, sind Objekte
- Primitive Werte → string, number, bool, null, undefined
- Ein Object ist eine Kollektion aus benannten Values
- Objekteigenschaften können primitive Werte, andere Objekte oder Funktionen als Werte haben

Objekte 2

EGOS!® the education company

- Objekte können erzeugt werden durch
  - Objekt literal → { }
  - „new“ Keyword → new Object()
  - Mit einem selbst erstellten Objektkonstruktor
  - Object.create()
- Objekte sind veränderbar und werden adressiert durch Referenz

```
const person = {
  firstName:"John",
  lastName:"Doe",
  age:50, eyeColor:"blue"
}

const x = person;
x.age = 10;       // Will change
both x.age and person.age
```

Objekte 3

EGOS!® the education company

- objectName.property → person.age
- objectName[„“] → person[age]
- objectName[expression] → x = „age“; person[x]
- For…in Loop → for(let person in family){ }
- Neue Eigenschaften hinzufügen
  - person.height = 1.90;
  - Object.defineProperty(obj, „propName“, propValue)
- Eigenschaften löschen
  - Delete person.age

Objekte 4

EGOS!
the education company

- Konstruktoren
  - ```
    function Person(first, last, age, eye) {
      this.firstName = first;
      this.lastName = last;
      this.age = age;
      this.eyeColor = eye;
    }
    ```
    ```

Object Prototype

EGOS!
the education company

```
function Person(first, last, age, eyecolor)
{
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

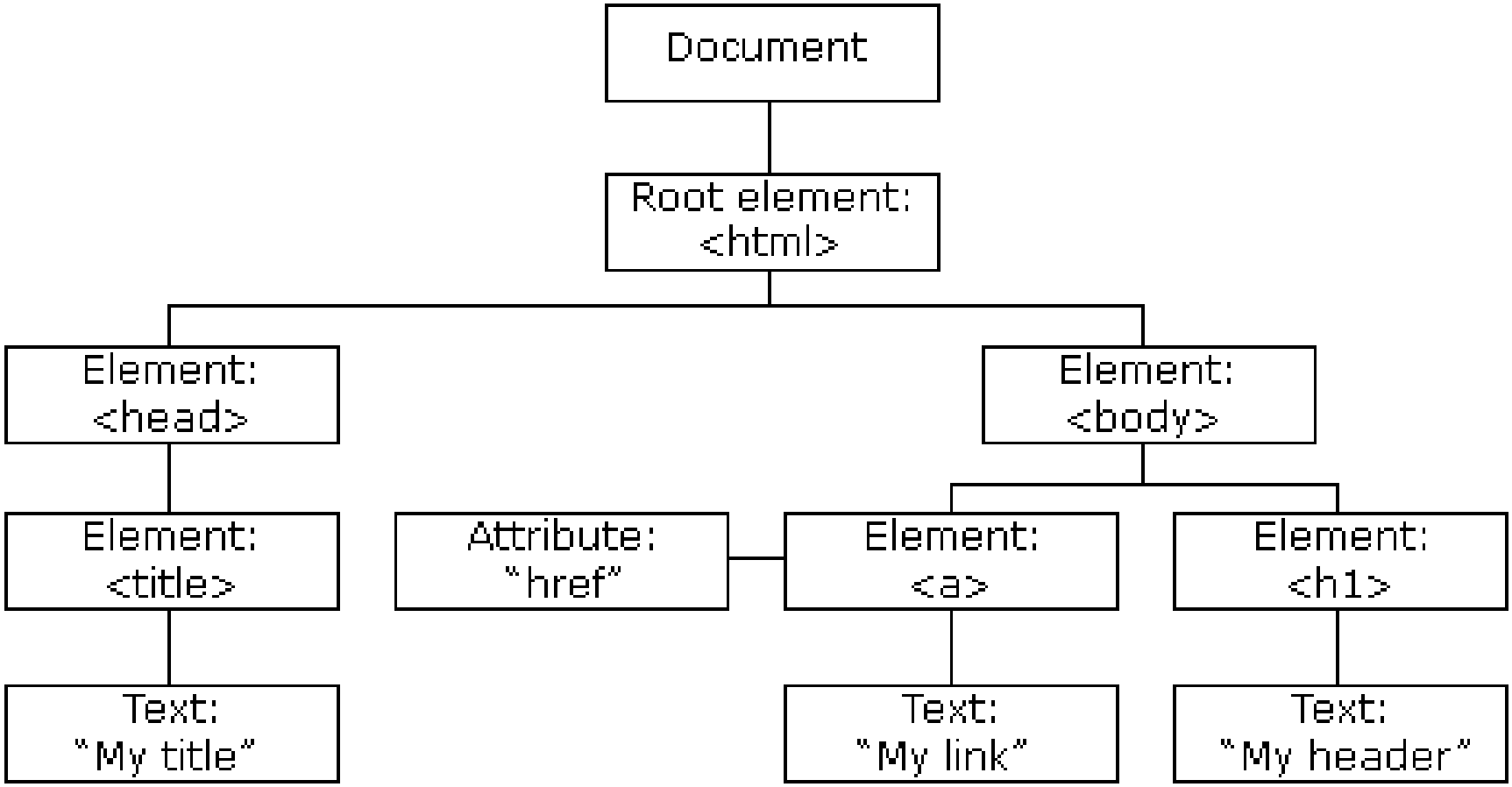
Person.prototype.nationality = "English";
```

Klassen

EGOS!
the education company

```
class ClassName1 {
  constructor() { ... }
  method_1() { ... }
  method_2() { ... }
  method_3() { ... }
}
class ClassName2 extends ClassName1 {
  constructor() { super(); ... }
  method_1() { ... }
  method_2() { ... }
  method_3() { ... }
}
```

DOM


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

DOM und JS

EGOS!® the education company

- JavaScript kann alle HTML-Elemente auf der Seite ändern
- JavaScript kann alle HTML-Attribute auf der Seite ändern
- JavaScript kann alle CSS-Styles auf der Seite ändern
- JavaScript kann vorhandene HTML-Elemente und -Attribute entfernen
- JavaScript kann neue HTML-Elemente und Attribute hinzufügen
- JavaScript kann auf alle vorhandenen HTML-Ereignisse in der Seite reagieren
- JavaScript kann neue HTML-Ereignisse auf der Seite erstellen

Kurz gesagt

EGOS!
the education company

- **Das HTML-DOM ist ein Standard zum Abrufen, Ändern, Hinzufügen oder Löschen von HTML-Elementen.**

Document Object accessors 1

EGOS!® the education company

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Document Object accessors 2

EGOS!® the education company

Property	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
Method	Description
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Document Object accessors 3

EGOS!®
the education company

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Document Object accessors 4

EGOS!®
the education company

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

HTML DOM properties

EGOS!® the education company

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Deprecated	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Obsolete.	3
document.embeds	Returns all <embed> elements	3
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

DOM Events

EGOS!
the education company

- `element.addEventListener(event, function, useCapture);`
- Events die erkannt werden
  - Click
  - Mouseover
  - Mousedown
  - Resize
  - Scroll
  - Load
  - .....

BOM

EGOS!
the education company

- Browser Objekt Model
- Erlaubt es das JavaScript mit dem Browser interagiert
- Window Object
  - window.open() - open a new window
  - window.close() - close the current window
  - window.moveTo() - move the current window
  - window.resizeTo() - resize the current window

Window.location

EGOS!® the education company

- window.location.href gibt die href (URL) der aktuellen Seite zurück
- window.location.hostname gibt den Domänennamen des Webhosts zurück
- window.location.pathname gibt den Pfad und den Dateinamen der aktuellen Seite zurück
- window.location.protocol gibt das verwendete Webprotokoll zurück (http: oder https:)
- window.location.assign() lädt ein neues Dokument

Window.history

EGOS!® the education company

- history.back() - dasselbe wie beim Zurückklicken im Browser
- history.forward() - dasselbe wie beim Vorwärtsklicken im Browser

Window.navigator

EGOS!® the education company

- Navigator.cookieEnabled – wenn cookies aktiviert sind „true“
- Navigator.appName – Browsername ;)
- Navigator.appCodeName – Codename Browser
- Navigator.product – Browser Engine
- Navigator.language – Browser Sprache

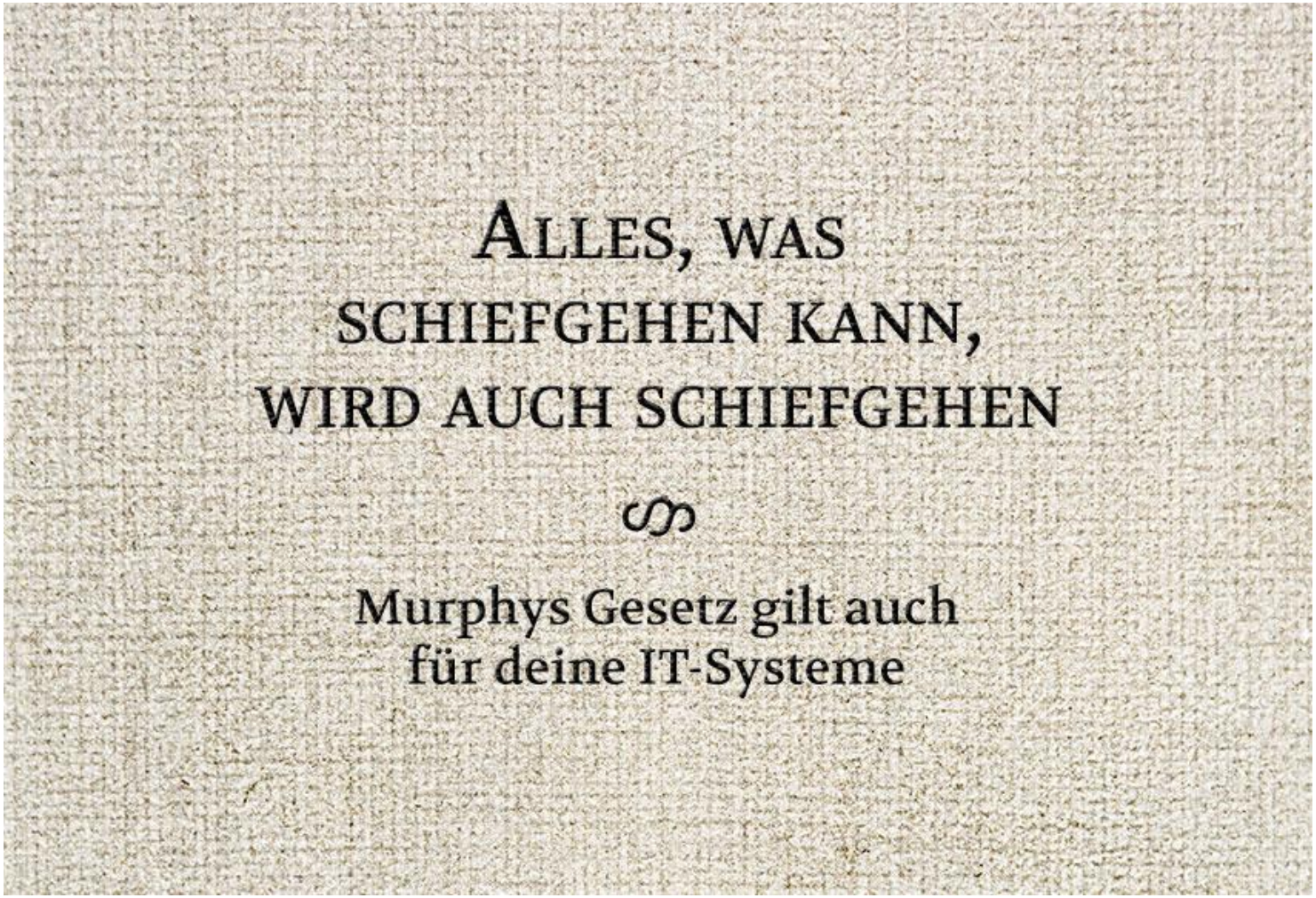
Cookies

EGOS!
the education company

```
document.cookie = "username=John Doe;
expires=Thu, 18 Dec 2013 12:00:00 UTC;
path=/";
```

Debugging

EGOS!
the education company



Debugging

EGOS!®
the education company

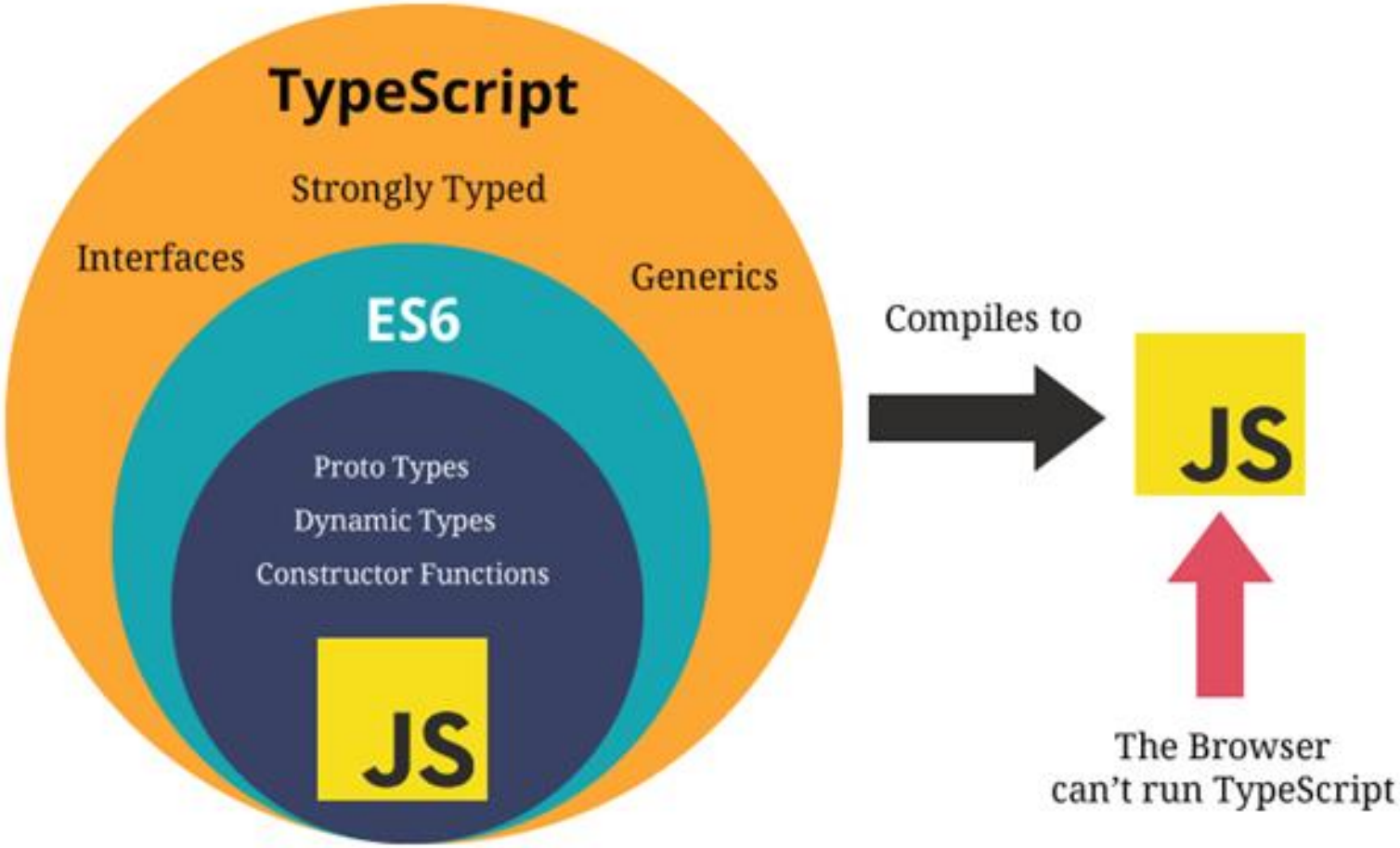
- Console.log()
- Breakpoints
- Debugger keyword

Helpful Links

EGOS!® the education company

- https://www.w3schools.com/jsref/default.asp

Typescript


TypeScript
Strongly Typed
Interfaces
Generics
ES6
Proto Types
Dynamic Types
Constructor Functions
JS
Compiles to
JS
The Browser
can't run TypeScript

- TypeScript ist eine syntaktische Obermenge von JavaScript, die statische Typisierung hinzufügt.
- Dies bedeutet im Grunde, dass TypeScript Syntax über JavaScript hinzufügt, sodass Entwickler Typen hinzufügen können.

Und warum TS

EGOS!
the education company

- JS Loosely typed language
- Oft schwierig zu verstehen wann welche Art von Daten verwendet werden
- In JS haben Funktionsparameter und Varaiblen keine Infos
- TS kann Datentypen angeben und Fehler melden wenn Typen nicht übereinstimmen
- Z.B. wenn erwarteter Wert Zahl aber es wird ein String übergeben wird in TS ein Fehler geworfen. In JS NICHT

Was geht in TS

EGOS!
the education company

- Object Types
- Enums
- Aliases & Interfaces
- Union Types
- Definition des Return Types einer Funktion
- Casting
- Klassen mit Sichtbarkeiten und Typen
- Implements, Extends, Override
- Abstakte Klassen
- Und und und

How to integrate TS into your JS Project

EGOS!® the education company

- https://www.freecodecamp.org/news/how-to-add-typescript-to-a-javascript-project/