Javascript einbinden

Head : <script src =”code.js” **defer** ></script>
Body : at the very bottom

Element aufrufen

document.querySelector vs document.querySelectorAll
- Eine Datei die du verbindest
- All = mehrere, dh du musst es später **durchloopen** !!

Inhalt ändern oder updaten

.innerText = ändere den Inhalt innerhalb des elements // zB “This is a new paragraph”
.innerHTML = ändere den inhalt des htmls // z.b This is a new element
---
Element erstellen und im html hinzufügen bzw entfernen

create new elements : document.createElement (“div”) // div, span, small, h1 , etc
append the created element = .appendChild(container) // where it belongs (container muss angegeben sein)
remove an element = .remove()

Beispiel wo wir das anwenden

```
const shoppingList = [
  “Apples”,
  “Bananas”,
  “Mangos”,
  “Cola”,
  “Redbull”,
  “Coffee”,
  “Rice”,
  “Bread”,
];

document.querySelector(“.shoppingList”).innerHTML = “”;
```

```
shoppingList.forEach((item) => {
  const li = document.createElement(“li”);
  li.setAttribute(“class”, “list-item”);
  li.innerText = item;
  document.querySelector(“.shoppingList”).appendChild(li);
});
```

das ist eine *LISTE* mit *MEHREREN* items, wir wissen dass wir **durchloopen** müssen um jedes einzelne aufzurufen

anstatt for(var i=0; i < x; i++)
benutzen wir **.forEach() =>**

für jedes item (apple,banana etc) erstellst du eine const und gibst ihr die class “list-item”
danach setzt du den inhalt deiner const, was das item selbst ist

danach fügst du es einfach hinzu

Ein Beispiel wo wir bei jedem click eine neue box erstellen und zeigen die wievielte es ist

wir brauchen dafür einen trigger, es kann ein div sein, ein link, ein bild idk nimm was willst

```
const trigger = document.querySelector(".trigger"); // was auch immer es ist, es hat diese class

let counter = 0; // du brauchst einen "platz" wo du die nummer speicherst, und du fängst bei null an
```

Wir erstellen eine funktion die sagt was beim CLICK passiert

```
trigger.addEventListener("click", function (e) {
  counter++; // bei jedem click +1, da könnte auch counter = counter +1 stehen
  console.log(e); // wir testen ob es geht
```

Bei jedem click erstellen wir eine box

```
  const box = document.createElement("div");
  if (counter % 2 == 0) {
    box.classList.add("even");
  }
  box.setAttribute("class", "box"); // setz eine class damit du es im css stylen kannst
                                    // es hätte auch box.classList.add("box") stehen können

  box.innerText = counter; // Der inhalt meines erstellten divs ist der counter (meine zahl)
```

Wir haben bei jedem click ein div erstellt und ihm den inhalt gegeben der anzeigt die wievielte box das ist ABER wir müssen sie noch in unsere seite einfügen, sonst "existiert" sie ja nicht...

```
  document.querySelector(".container").appendChild(box);
});
```

Wie können wir erstellte elemente im js stylen ?

```
.classList.add ("sth")  // wir geben dem element eine class und stylen im css
.setAttribute("style", 'color: red') // wir stylen im js selbst
```

Speichere den Input des Users und verwende es

```
<body>
   <h1>Greeter App</h1>
   <form>
     <label for="name">Your Name</label>
     <input type="text" name="name" id="name" />
     <button id="send" type="submit">Send</button>
   </form>
 </body>


const button = document.querySelector("#send"); // unser trigger

button.addEventListener("click", (e) => {                    // (hier könnte auch foo stehen)
                                                             // in userem fall bedeutet e einfach event
  e.preventDefault();
```

Wir müssen eine const erstellen wo wir den input "speichern" damit wir es verwenden können

```
  const inputValue = document.querySelector("#name").value; //.value ist der inhalt
```

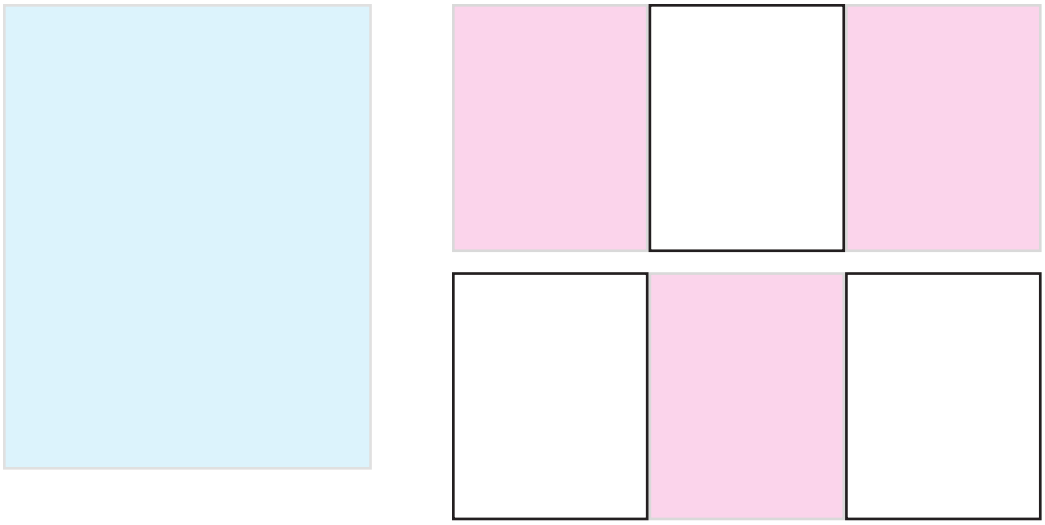
Was machen wir mit dem namen? wir wollen den user mit seinem namen ansprechen -
also erstellen wir doch ein <p> wo wir das machen

```
  const nameOutput = document.createElement("p"); // p erstellt

  nameOutput.innerText = "Hello ${inputValue}!"; // wir setzen den inhalt des <p>
```

Wo wird unser satz angezeigt ?

```
  document.querySelector("form").after(nameOutput);
});
```

Erstellen wir eine simple gallery



Wenn ich auf eines der kleineren images clicke dann wird es links gross angezeigt

wir werden ganz einfach die SRC zuweisen, das ist die einfachste lösung

```
<div class="container">
    <div class="main-img">
      <img src="img/cat1.jpg" alt="" />
    </div>
```

Das ist das grosse bild links

```
    <div class="images">
      <img src="img/cat1.jpg" alt="" />
      <img src="img/cat2.jpg" alt="" />
      <img src="img/cat3.jpg" alt="" />
      <img src="img/cat4.jpg" alt="" />
      <img src="img/cat5.jpg" alt="" />
      <img src="img/cat6.jpg" alt="" />
      <img src="img/cat7.jpg" alt="" />
    </div>
  </div>
```

die rechte seite. im css kannst du ganz einfach grid benutzen

grid-template-columns : repeat (3, 1fr)

Du musst ALLE images auswählen, deshalb ALL

```
const images = document.querySelectorAll(".images
.img"); //wir wissen dass wir loopen müssen

const current = document.querySelector(".main-img
.img"); // nur eins, kein loop

images.forEach((img) => {
  img.addEventListener("click", imageClick);
});
```

für jedes img (forEach (img) es kann auch foo sein) fügst du ein event zu, click

bei jedem click soll die erstelle funktion imageClick ausgelöst werden

```
function imageClick(e) {  //hier könnte auch lalala stehn
  current.src = e.target.src;
}
```

//die current, also jetzige src wird "umgeschrieben" wir ändern sie bei jedem click zu der src die das geklickte bild hat :)

jQuery - das ist eine library, dh du musst sie einbinden

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
```

Sobald DOM geladen hat, wird die funktion ausgeführt

```
$(document).ready(function () { });
```

Wähle dein element aus und style es cause why not

```
$(".foo").css("border", "solid 1px black")
$("#foo").css("border", "solid 1px black")
```

Wie können wir ein element am anfang setzen, am ende, inhalt ändern etc ?

```
.append (am ende + )
.prepend (am anfang +)
.remove (versteht sich von selbst)
.html (wie innerHtml) - ("<h2>hello world</h>")
.text (wie innerText)- ("<tag>this is a tag </tag>")
```

Ein Beispiel wo wir ein event setzen. Wenn du den button clickst wird eine function ausgelöst.
die function heisst setColor

```
$(document).ready(function(){
    $("button").click(function(){
        setColor($('input').val())
    })
    function setColor (col) {   // col = color , da hätte auch banana stehen können
        $(".box").css("background-color",col);
        $(".text").css("color",col);
        $(".circle").css("background-color",col);
    }
})
```

Für ein Hover effect brauchen wir zwei functions weil etwas passiert wenn du mit der maus drüber bist und dann etwas passieren muss wenn du deine maus entfernst

.hover(function(){}, function (){}) // machs dir einfach und schreibe das zuerst aus, danach öffne die klammern

```
$(document).ready(function(){

    $(".low").hover(function(){
        $( this ).append("<span id='wait'> ... this can wait</span>");  //erste funktion, mouseover
    }, function(){
        $("span").remove(); // zweite funktion, mouseleave
    })
});
```

Paar nützliche jQuery effects

.hide () // vergiss nicht dass du ein timing setzen kannst in ms, zb 2000 = 2seconds
.show()
.slideUp()
.fadeIn()
.fadeOut()
.toggle()

Ein Beispiel wo wir paar effects verwenden

```
$(document).ready(function () {  // nachdem DOM geladen ist ....
$("#hide-p").click(function () { // element mit dieser id bekommt ein click-event....
        $("p").hide("1000", function () {  // p verschwindet nach 1s und es wird eine function ausgeführt..
          $(this).removeClass("lookAtMe");  // ..wo das aktive element die class "lookAtMe" weg bekommt
        });
       });

       $("#animate-p").click(function () {  // element mit dieser id bekommt ein click event....
         console.log("hi");  // testen obs geht
         $("p").show(750, function () {  // beim click wird mein p gezeigt nach 0.75s und die function...
           // this = DOM element which has just finished being animated
           $(this).addClass("lookAtMe");  // ...function wird ausgeführt wo mein element die class bekommt
         });
       });
     });

 // TOGGLE effect
   $(document).ready(function () {

       $(".tog-list").click(function () {  // mein element mit dieser class bekommt ein click event...
         $("ul").fadeToggle(1250);  // mein ul wird getogglet mit dem fade effect, 1,25 s dauer
       });

 // mein hover effect
       $(".fade-box").hover(  // mein element mit dieser class bekommt den hover event...
         function () {  // ... hover hat diese function beim erstem drüber gehen
           $(".box").fadeOut("slow", function () {  // meine box wird langsam ausgefaded...
             console.log("Faded Out");  // schauen obs geht
           });
         },
         function () {  // ...hover hat diese function beim verlassen
           $(".box").fadeIn("fast", function () {  // box wird schnell eingefaded...
             console.log("Faded back in");  // test
           });
         }
       );
     });
```

Benutzen wir animate um eigene effects zu erstellen

```
$("#box").animate(  // animate damit du selber entscheidest was passieren soll
          {
            height: 900,
            width: 600,
          },
          600,  // nach 0.6s
          function () {
            console.log("Boxy is fat now");  // in diesem fall wird die höhe und breite geändert
          }
        )
```

RECAP

```
    $(document).ready(function () {  // nachdem mein DOM geladen ist....
        // wenn ich mit der maus auf meine box geh wird die function "coordinatesDisplay" ausgelöst
       $(".box").on("mousemove", coordinatesDisplay);
// wenn ich über meine box hover dann werden diese zwei functions ausgelöst
// wir wissen dass bei einem hover zwei function gebraucht werden !
       $(".box").hover(mouseEnter, mouseLeave);

       function coordinatesDisplay(evt) {  // function die die coordinates anzeigt
         //console.log("x: " + evt.pageX + " y: " + evt.pageY);  // zeigt x und y an
         console.log(evt);
         $("#coor").text("Mouse X: " + evt.pageX + " Mouse Y: " + evt.pageY);
// unser element mit dieser id bekommt den inhalt von den coordinates
       }

       function mouseEnter() {  // hover - wenn du drüber gehst
         //console.log("Mouse Enter")
         $("#info").text("Mouse is in the Box");
         $(".box").css("background-color", "lightcoral");
       }
       function mouseLeave() {  // hover - wenn du es verlässt
         //console.log("Mouse is out of the Box")
         $("#info").text("Mouse is out of the Box");
         $(".box").css("background-color", "lightblue");
       }
     })
```

SLIDER - wieso passiert was

overflow

```
<body>
  <div id="slideshow">
// wir brauchen etwas um vor und zurück clicken zu können
    <a href="#" class="button-weiter">&gt;</a>  // wenn wir das clicken dann gehts eins weiter
    <a href="#" class="button-zurueck">&lt;</a>  // wenn wir das clicken eins zurück
    <ul>
      <li>Slide 1</li>
      <li style="background: #aaa">Slide 2</li>
      <li>Slide 3</li>
      <li style="background: #aaa">Slide 4</li>
    </ul>  // das sind unsere elemente im slider, zb bilder
  </div>

// etwas wo der user clicken kann um den automatischen effect auszulösen
  <div class="slider-option">
    <input type="checkbox" id="toggle-animation" />
    <label for="toggle-animation">Autoplay Slider</label>
// der name sagt schon dass es getogglet wird
  </div>
</body>
```

Wir müssen es stylen

```
#slideshow {
  position: relative;
  overflow: hidden;
  margin: 20px auto 0 auto;
  border-radius: 40px;
}
#slideshow ul {
  position: relative;
  height: 200px;
  list-style: none;
}
/* Slides */
#slideshow ul li {
  position: relative;
  display: block;
  float: left;
  width: 500px;
  height: 300px;
  background: #ccc;
  text-align: center;
  line-height: 300px;
}
```

// wir geben unserer gallery eine bestimmte grösse und da wir eine liste haben von mehreren bildern die nebeneinander angezeigt sind (der default fluss ist von links nach rechts) müssen wir, das was nicht in meine vorgabe passt verstecken, also overflow hidden

// positioning ist hier wichtig

// hier geben wir an wie hoch und wie breit unsere bilder sind - 500 auf 300 px

text-align : center richtet unsere images in die mitte

(im beispiel warens einfache slides mit text deshalb line-height)

also nochmal:
wir haben einen container mit der id #slideshow wo wir unseren inhalt setzen können. da wir mehrere elemente angereiht haben müssen wir den overflow verstecken.

in unserem container haben wir dann unseren inhalt in form von einer ul mit li

wir geben alles unseren elementen eine positioning da wir es ja positionieren müssen :)

unsere li müssen eine bestimmte grösse haben, die geben wir

```
a.button-weiter,
a.button-zurueck {
  position: absolute;
  top: 40%;
  z-index: 999;
  display: block;
  padding: 4% 3%;
  width: auto;
  height: auto;
  background: #2a2a2a;
  color: white;
  text-decoration: none;
  font-weight: 600;
  font-family: sans-serif;
  font-size: 1.2rem;
}

a.button-zurueck {
  border-radius: 0 2px 2px 0;
}
```

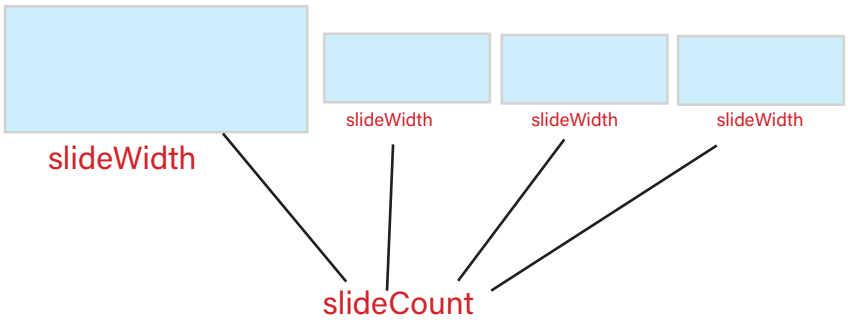
// stylen von den buttons
wir geben denen die position absolute um sie frei setzen zu können

```
a.button-weiter {
  border-radius: 2px 0 0 2px;
  right: 0; // andere seite
}

.slider-option {
  position: relative;
  margin: 10px auto;
  width: 160px;
  font-size: 1.1rem;
}
```

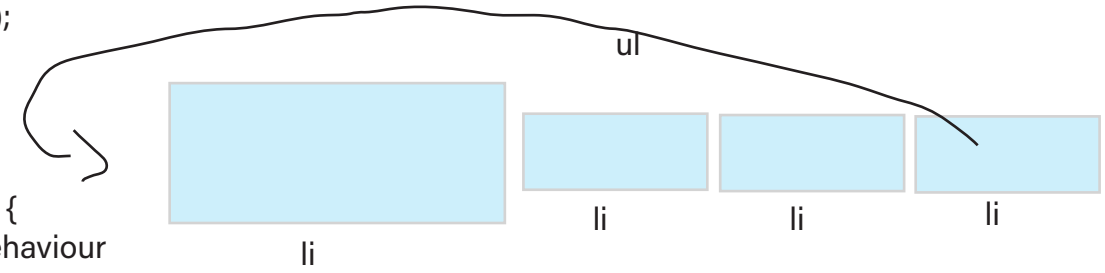
THIS IS THE FUN PART

```
$(function () { // nachdem unser DOM geladen hat wird das ausgeführt
//  wir definieren alle unsere elemente die wir brauchen
  // Control Elements
  let prevBtn = $("a.button-zurueck"), // button für zurück
    nextBtn = $("a.button-weiter"), //  button für weiter
    checkbox = $("#toggle-animation"); //  unser click für automatisch
//  elemente im slider
  let slider = $("#slideshow"), //  unser container
    sliderUl = slider.children("ul"), // vom container das child ul
    slides = sliderUl.children("li"), //  vom ul die childer aka li
    slideCount = slides.length, //  die anzahl der elemente ist die grösse der liste (li.length)
    slideWidth = slides.width(), //  die breite der elemente ist die vorgegebene width ( im css unser width)
    slideHeight = slides.height(), //  die höhe aus unserem css
    sliderUlWidth = slideCount * slideWidth, // achtung mathe :
    sliderInterval;

  // Shrink the slide, making it as big as a slide
   slider.css({ // es soll die grösse unserer li haben
    width: slideWidth,
    height: slideHeight,
  });
  // fit the row of slides
  sliderUl.css({ //unser ul im container
    width: sliderUlWidth,
    marginLeft: -slideWidth,
  });
  // Take the last slide and prepend it to the row of slides
//das letzte wird nach vorne gesetzt mit prepend
  slides.last().prependTo(sliderUl);
```

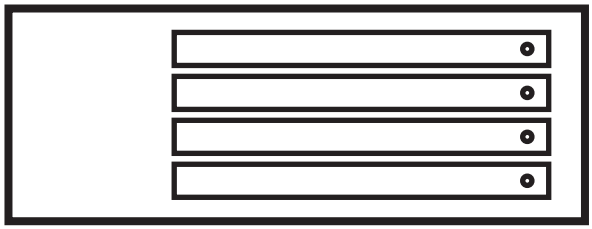
in diesem beispiel 4 x 500px



```
// 2: Events die wir definieren werden

 prevBtn.on("click", moveLeft); // button on click wird diese function ausgeführt
 nextBtn.on("click", moveRight); // button on click wird diese function ausgeführt
 checkbox.on("click", onSliderToggle); // box on click wird diese function ausgeführt

// 3: Functions die wir jetzt setzen für unsere clicks

 function moveLeft(e) { // unser button der nach links zeigt
   // Prevent Default Behaviour
   e.preventDefault();
   sliderUl.animate( // on click kommt dieser effect den wir selbst setzen mit animate
     {
         left: +slideWidth, // angezeigtes bild wir um ein bild versetzt, da wir overflow hidden haben sehen
                                                                  wir das erste bild nicht mehr
     },
     200, // 0.2 s
       function () { //wir brauchen das damit die bilder net einfach verschwinden sondern in der liste wied-
                                                                  er richtig gesetzt werden
       // We take te last li in the ul and prepend it to the ul
       $(this).children("li").last().prependTo(sliderUl);
       // We reset the left value which we changed in the animation
       $(this).css("left", "");
     }
   );
 }

 function moveRight(e) {
   // Prevent Default Behaviour
   if (e) {
     e.preventDefault();
   }

   sliderUl.animate(
     {
       left: -slideWidth,
     },
     200,
     function () {
       // We take te last li in the ul and prepend it to the ul
       $(this).children("li").first().appendTo(sliderUl);
       // We reset the left value which we changed in the animation
       $(this).css("left", "");
     }
   );
 }

 function onSliderToggle(e) {
   // Check if checkbox is checked
   let check = $(e.currentTarget);
   if (check.is(":checked")) {
     sliderInterval = setInterval(moveRight, 3000); // alle 3s
   } else {
     // Stop interval
     clearInterval(sliderInterval);
   }
 }
});
```



Accordeon mit js



```
<div class="container">
    <div class="left-panel"> // einfach die linke seite mit text
      <h2>Questions and ansswers about login</h2>
    </div>
    <div class="right-panel"> // der eigentlich wichtige part
      <!--Items in the accordion  (6)-->
      <div class="item"> // erstes item, mit infos drinne
        <div class="item-title">
          <p class="title-text">Do I have to allow the use of cookies?</p>
          <i class="fas fa-plus"></i> // das brauchen wir für unser click event
        </div>
        <div class="item-info">
          <p class="info-text">
            Yes, in order to use PAGENAME, you must allow the use of cookies
            in your browser.<br />
            <br />See also: What is a cookie and what does it do?
          </p>
        </div>
      </div>
    </div> //right panel end
  </div> //container end
```

kopiere es so viel du brauchst

```
.container {
  box-shadow: 2px 2px 8px
rgb(100, 100, 100);
  border-radius: 1rem;
  width: 900px;
  display: grid;
  grid-template-columns: 1fr 2fr;
  background: white;
  padding: 3rem;
}

.left-panel h2 {
  font-size: 2rem;
}

.item {
  border: solid 1px #ccc;
  padding: 1rem;
  margin: 5px 0;
  border-radius: 5px;
}

.item-title {
  display: flex;
  justify-content: space-between;
}

.title-text {
  font-weight: 600;
  display: flex;
  align-items: center;
}

.info-text {
  margin-top: 1rem;
}

.item-info {
  display: none;
}

.fas {
  border-radius: 50%;
  background: rgb(236, 236, 236);
  color: rgb(255, 143, 199);
  width: 35px;
  height: 35px;
  display: flex !important;
  justify-content: center;
  align-items: center;
}
```

```
$(function () {
  $(".fas").click(function () { // unser i, plus icon bekommt ein click event
    //wir setzen ein reset für alle die nicht geklickt worden sind weil es ja sonst jedes aktivieren würde
    $(".item-info").not($(this).parent().siblings()).slideUp();
    //alle zeigen plus bis auf das welches geklickt worden ist -> es ist ein minus
    $(".fa-minus").not($(this)).removeClass("fa-minus").addClass("fa-plus");
    //toggle classes
    $(this).toggleClass("fa-plus");
    $(this).toggleClass("fa-minus");
    // slide down von dem was du geclickt hast
    $(this).parent().siblings().slideToggle(); //es soll ja den inhalt vom geklicktem
anzeigen deshalb verwenden wir parent and siblings , sonst würde es jedes
aufrufen
  });
});
```

```
<!-- Container for tab system -->
<div class="tab-container"> // das oberste wo wir taben werden (hier drei)
  <!-- Clickable Tabs -->
  <ul class="tab-auswahl"> //hier sind unsere drei tabs
    <li class="active">Tab #1</li>
    <li>Tab #2</li>
    <li>Tab #3</li>
  </ul>
  <!-- Tab content -->
  <div class="tab-inhalt"> //der untere teil mit pro tab eine info
    <div>
      Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolore, quis
      temporibus.
    </div>
    <div>Lorem ipsum, dolor sit amet consectetur adipisicing elit.</div>
    <div>
      Lorem ipsum, dolor sit amet consectetur adipisicing elit.
    </div>
  </div>
</div>
```

```
.tab-container {
  margin: 60px auto;
  width: 70%; // 70% von der länge vom container
  max-width: 500px;
}

.tab-auswahl li {
  position: relative;
  background: #fff;
  color: rgb(73, 73, 73);
  display: inline-block;
  padding: 20px 40px;
  opacity: 0.8;
  cursor: pointer;
  z-index: 0;
}

.tab-auswahl li:hover {
  color: #222;
}

.tab-auswahl li.active {
  color: #222;
  opacity: 1;
}
```

```
.tab-inhalt > div {
  background: #fff;
  width: 100%; // vom container selbst
  padding: 50px;
  min-height: 200px;
}

.line { //das wird später reinanimiert beim click
  position: absolute;
  width: 0;
  height: 7px;
  background: hotpink;
  top: 0;
  left: 0;
}
```

```
$(function () {
// zuerst definieren wir alle unsere nötigen variablen
  const wrapper = $(".tab-container"); // der obere teil mit den drei tabs
  const allTabs = wrapper.find(".tab-inhalt > div"); // der dazugehörige div mit der info
  const tabMenu = wrapper.find(".tab-auswahl > li"); // die dazugehörige li
  const line = $('<div class="line"></div>').appendTo(tabMenu); // die linie die beim click erscheint

  // wir verstecken alle divs die nicht aufgerufen sind, im default zeigts einfach das aller erste an
also verstecken wir alle die nicht first sind
  allTabs.not(":first-of-type").hide(); //alle die nicht als erstes stehen
  // Change the width of the first lines to 100% when page is loaded
  tabMenu.filter(":first-of-type").find(":first").width("100%");

//wir setzen das für ALLE unsere tabs
  // set the data attribute of the li and the div
  tabMenu.each(function (i) {   //wie forEach, also für jedes element setzen wir eine function
    $(this).attr("data-tab", "tab" + i); //attr () value of an attribute
  });

  allTabs.each(function (i) {
    $(this).attr("data-tab", "tab" + i);
  });

  tabMenu.on("click", function () {
    let dataTab = $(this).data("tab");
    let getWrapper = $(this).closest(wrapper);
//class active bekommt der geklickte
    getWrapper.find(tabMenu).removeClass("active");
    $(this).addClass("active");

    //Reset the width of all lines
// nur active element bekommt die 100% line, alle anderen 0% weil sie ja nicht geklickt worden sind
    getWrapper.find(".line").width(0);
    // Animate the width of clicked line
    $(this).find(line).animate({ width: "100%" }, "fast"); // das ist die animation dazu

    // We reset all the content divs
//nur der inhalt zum zugehörigem tab soll angezeigt werden, alle anderen versteckt
    getWrapper.find(allTabs).hide();
    // show the tab content using it's corresponding data attribute
    getWrapper.find(allTabs).filter('[data-tab=${dataTab}]').show();
  });
});
```

Da wir für jeden tab, jeden plus und minus *eine funktion* setzen, löst es alle aus, da sie alle gleich aufgebaut worden sind.
Deshalb die resets, damit nur das, welches geklickt worden ist angezeigt wird