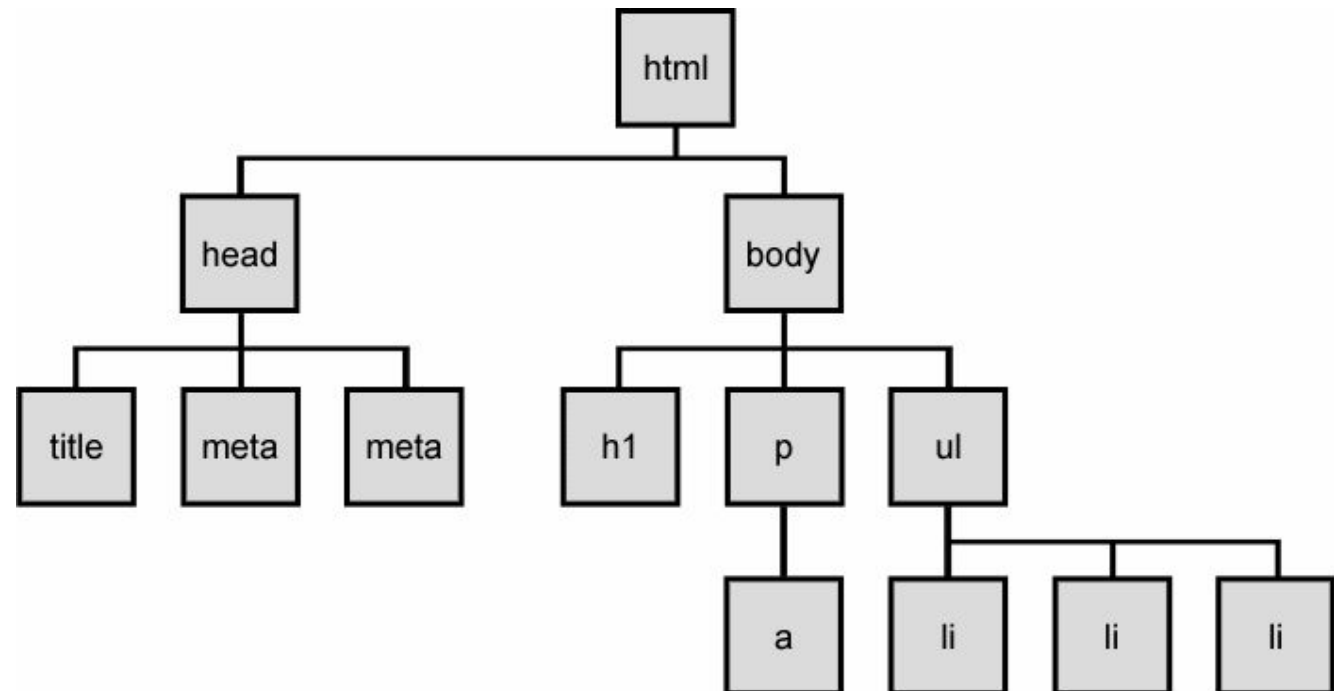


# Input elements and events



# Agenda

- How to use the input tag
- How to query for elements
- Event-driven programming
- Bad ways of getting input
- function
- parseInt()
- The text field
- Id
- label
- Button
- onclick
- innerHTML
- document.getElementById()
- document.getElementsByClassName()
- document.getElementsByTagName()

# Quick Functions Review

- How do we define a function in JS?

```
var funcName = function() {  
    console.log("this is a function");  
};
```

- How do we call a function in JS?

```
funcName();
```

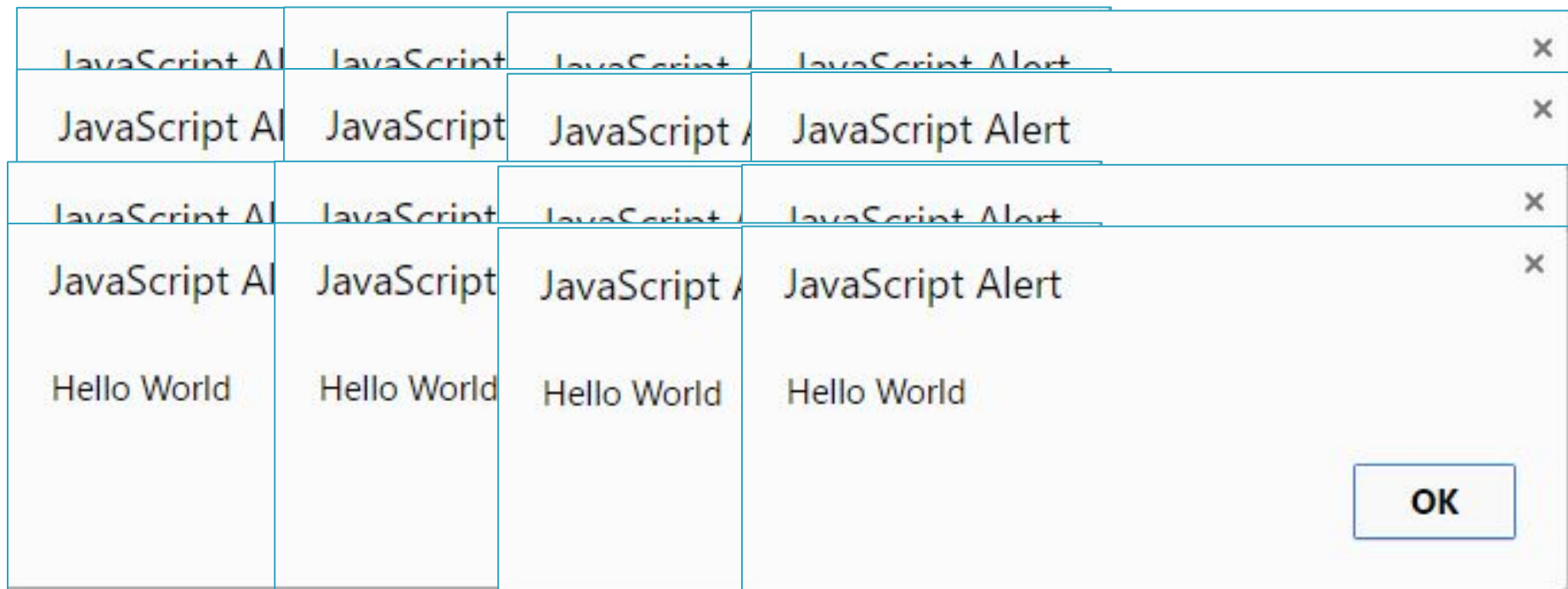
- How do we define a function with parameters?

```
var calcSum = function (a,b) {  
    return a+b;  
};  
  
console.log(calcSum(2,3));
```

# Input/Output □ I/O

We said that using alert and prompt are not the right ways to get input from the user.

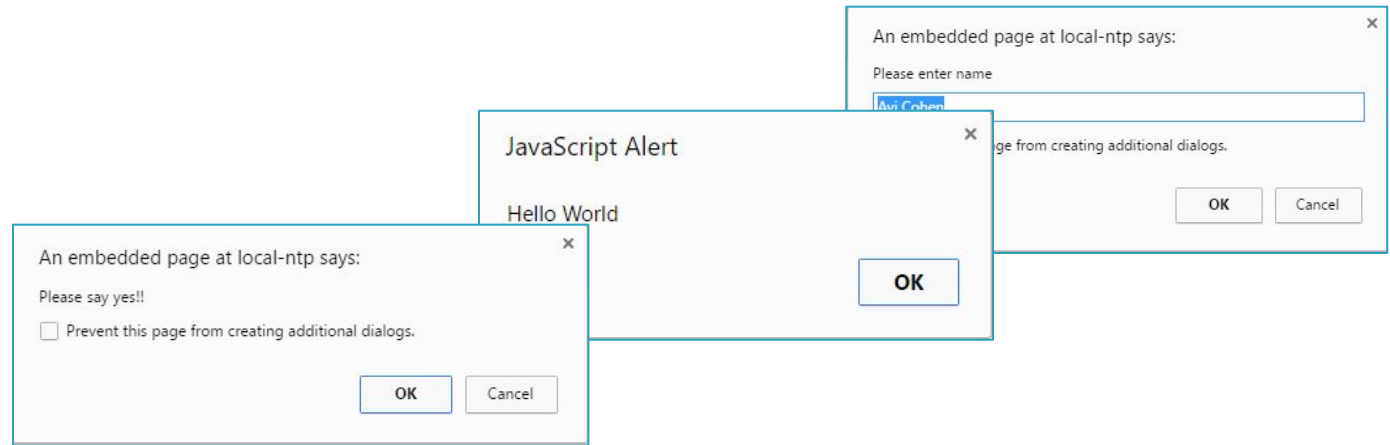
Before we see the right way, let's see the wrong ways of getting the user input



# Bad I/O

- There are 3 main types of “Bad ways of getting input from the user”

- Prompt
- Alert
- Confirm



This is not something we will use in our websites.  
**EVER!.** It annoys the hell out of users.

- It stops the user interface (UI) from doing anything else until the user is done answering (called ‘blocking code’)
- This means it creates very bad user experience (UX).
- **We will not use them again from now on!**

# Now, the right way

The `<input>` tag

# The `<input>` tag - types

Let's start from the `html`

We have a lot of input types:

- `button`
- `checkbox`
- `color`
- `date`
- `datetime`
- `datetime-local`
- `email`
- `File`

# The text field

```
<span>This is an input field </span>  
<input type = "text" />
```

This is a void tag

The input tag has an important attribute called "type" .

Using the same tag, we can create different input elements.

This is an input field



# The text field

- The user can click the field to get focus and add text



This is an input field

# The text field

- The text field can have a placeholder

```
<span>This is an input field </span>  
<input type="text" placeholder="email" />
```

This is an input field

# Placeholder

And like any other element it can be styled

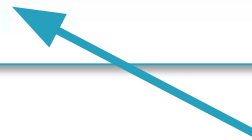
## HTML

```
<span>This is an input field</span>
<input type="text" placeholder="email"/>
```

## CSS

```
input {
  background-color: black;
  color: white;
  border: none;
}
```

This is an input field email



This is not white

We are setting the color of the user's text and not the placeholder's text...

# Placeholder CSS

Changing the placeholder color

```
::-webkit-input-placeholder {  
    color: red;  
}  
  
:-moz-placeholder { /* Firefox 18- */  
    color: red;  
}  
  
::-moz-placeholder { /* Firefox 19+ */  
    color: red;  
}  
  
:-ms-input-placeholder {  
    color: red;  
}
```

# Input value attribute

We can set the value of the field

```
input{  
    background-color:black;  
    color:white;  
    border:none;  
    font-size:20px;  
    font-family:Arial;  
}
```

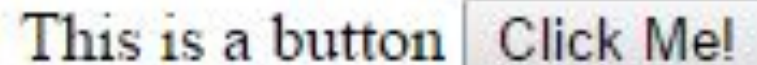
```
<span>This is an input field </span>  
<input type="text" value="pre entered text" />
```

This is an input field **pre entered text**

# Input type Button

This is what happens when we change the type to "button"

```
<span>This is a button| </span>  
<input type="button" value="Click Me!" />
```



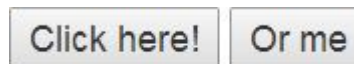
This is a button

# Buttons

A button can be:

1. An input with type button
2. A button tag

```
<input type="button" value="Or me" id="button2" />  
<button id="button1">Click here!</button>
```



# Input Types

There are more types for the <input> tag

```
<span>This is a checkbox </span><input type = "checkbox" /><br/>
<span>This is a radio button </span><input type = "radio" /><br/>
<span>This is a password field </span><input type = "password" /></br>
<span>This is a file selector </span><input type = "file" /></br>
```

This is a checkbox ☐

This is a radio button ☐

This is a password field

This is a file selector  Choose File No file chosen

This is a checkbox ☒

This is a radio button ☒

This is a password field .....

This is a file selector  Choose File No file chosen



# Questions?

```
console.log("Questions?");
```

# Id Attribute

So we have different types of inputs

This is great, but how do we identify the input?

We saw we can use different css selectors to identify a specific element

The best way to uniquely identify an element is to give it an id



# Id Rules

We'd want to identify an input field using a unique attribute



The ID of an element should be unique



The ID of an element cannot start with digits



It must not contain spaces



It is a good practice to separate words by a dash: my-id

or using camelCase: myId



# Labels

Labels are connected to input elements

- A label can describe what an input field expects to get
- You connect them using the "for" attribute in the label
- The for value will be the id of the input
- Bounding a label to an input is better semantics. Also by clicking a bound label the input will be selected

```
<label for="is-over-18">Are you over 18?</label>  
<input type="checkbox" id="is-over-18"/>  
<br/>  
<label for="first-name">First name</label>  
<input type="text" id="first-name" placeholder="Avi"/>  
<br/>  
<label for="last-name">Last name</label>  
<input type="text" id="last-name" placeholder="Cohen"/>  
<br/>  
<button>Click here!</button>
```



The screenshot shows the rendered HTML form. It contains four elements: a label 'Are you over 18?' followed by a checkbox, a label 'First name' followed by a text input field containing 'Avi', a label 'Last name' followed by a text input field containing 'Cohen', and a button labeled 'Click here!'.

# Questions?

```
console.log("Questions?");
```

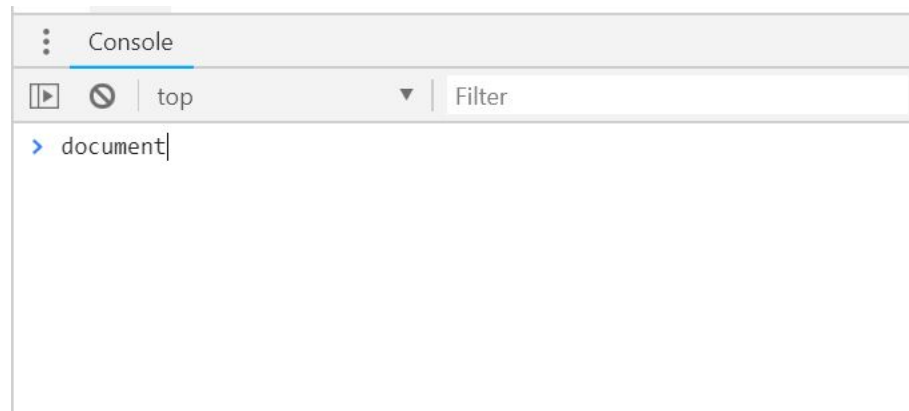
# Document Object

```
<span>This is an input field </span>  
<input type="text" value="pre entered text" />
```

This is an input field pre entered text

In order to get the value of a field with an id, we need to use javascript and access the document.

Let's type **document** in the console!



# Document

What is the document?

What is the type of document?

The document is an object (a javascript object)

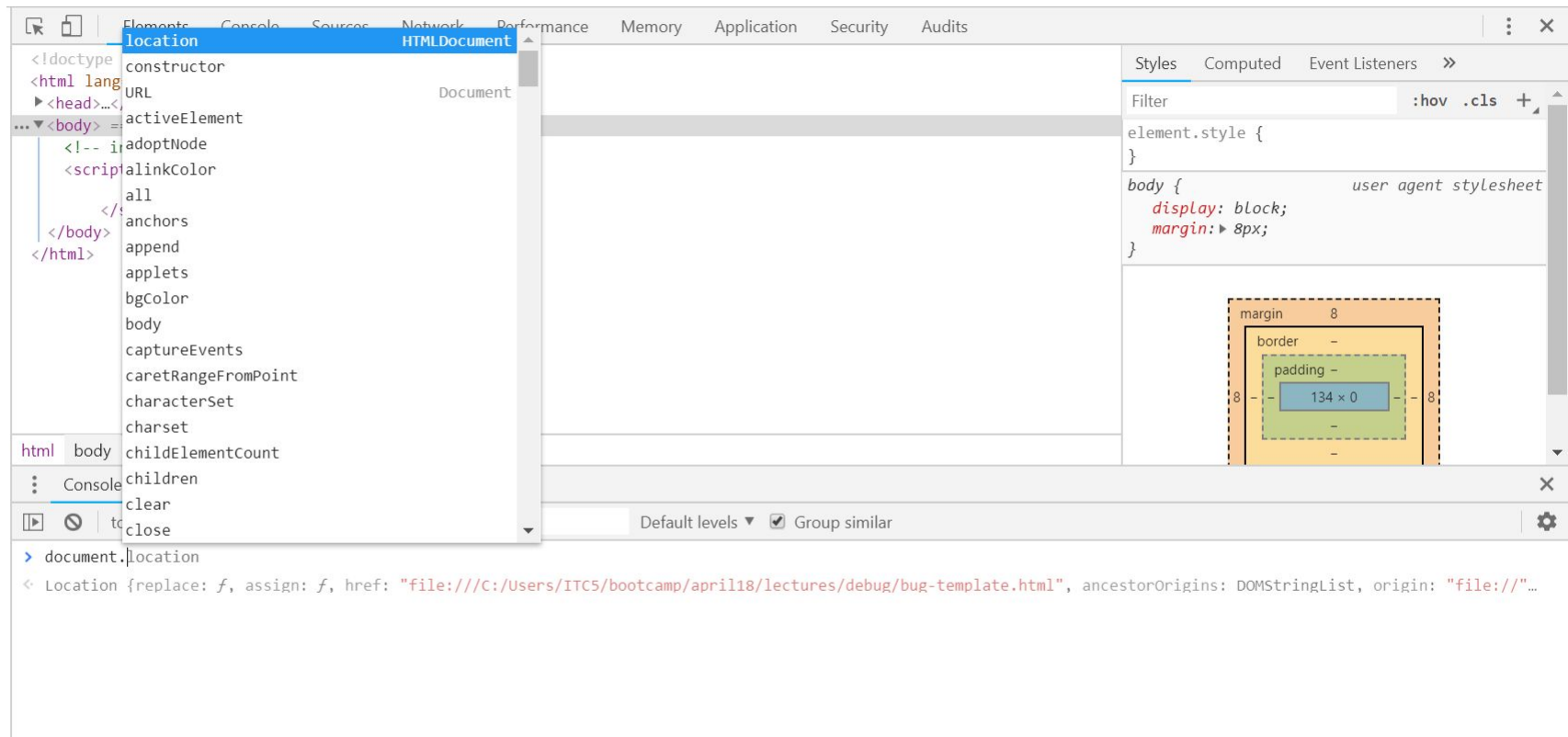
It is a global object that is known to the browser

What is inside the document?

Let's explore the document object in the console!

# Document

The document contains a lot of functions and attributes.





# Getting input from the user

Example:

```
<label>first name:</label>  
<input id="first-name" />  
<br/>  
<label>last name:</label>  
<input id="last-name">
```

first name:   
last name:

In order to get a value of an input we need to do:

```
var fname = document.getElementById("first-name").value;
```

# Getting input from the user

Example:

```
<label>first name:</label>
<input id="first-name" />
<br/>
<label>last name:</label>
<input id="last-name">
```

first name:

last name:

Let's break it down:

```
document.getElementById("id-name").value;
```

Document object

HTML element

The field value (String)

# Getting input from the user

```
<label>first name:</label>
<input id="first-name" />
```

document.getElementById("id").value

HTML element

What is this HTML we are getting back?

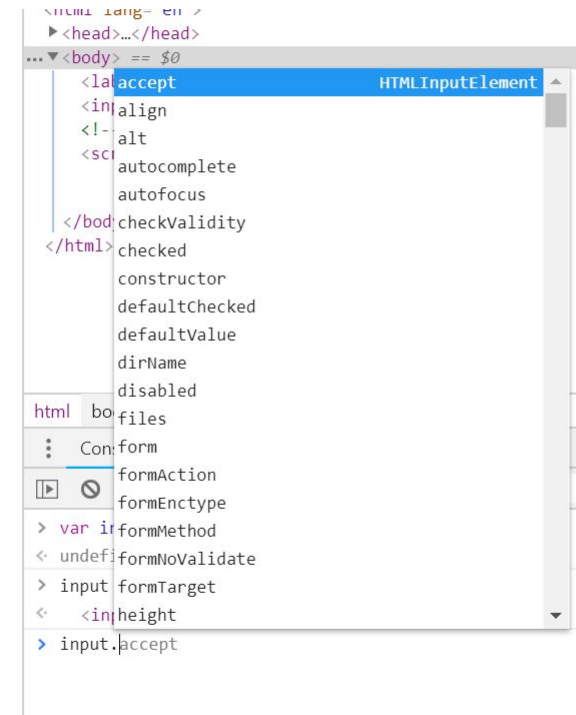
It is a javascript representation of the html element

It is a javascript **object** that contains properties and functions

```

> var input = document.getElementById("first-name")
< undefined
> input
< <input id="first-name">
> |

```



# Getting input from the user

Full Example:



```
<label>first name:</label>
<input id="first-name" />
<br/>
<label>last name:</label>
<input id="last-name">
```

first name:

last name:



```
<script>
  var fname = document.getElementById("first-name").value;
  var lname = document.getElementById("last-name").value;
  console.log("hello " + fname + " " + lname);
</script>
```

Reminder: The javascript can be written in the script tag or in a js file linked to an html file

# Getting input from the user

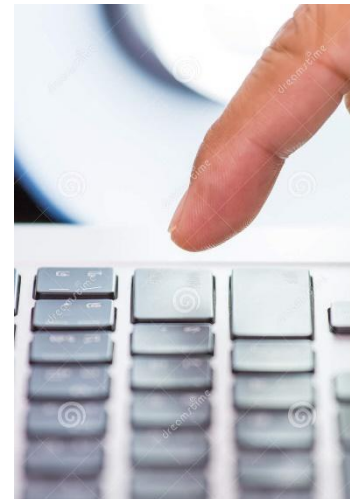
```
<script>  
  var fname = document.getElementById("first-name").value;  
  var lname = document.getElementById("last-name").value;  
  console.log("hello " + fname + " " + lname);  
</script>
```

- When will this code be executed?

Before the user entered his name...

- How can we change this?

Event driven programming!



# Event Driven Programing



In computer programming, event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs.

# Event Driven Programming

- In event driven programming we are binding (=connecting) an event to a specific function
- In this way the function will run only after the event happened.
- Luckily, we already learned how to write functions in JavaScript...



# Event Driven Programming

Back to our code

```
var sayMyName = function () {  
    var fname = document.getElementById("first-name").value;  
    var lname = document.getElementById("last-name").value;  
    console.log("hello " + fname + " " + lname);  
};
```

We will use a button, so the user will be able to tell us when he finished to enter his name



# Event Driven Programming

We only need to bind the event “user is clicking on the button” to the function sayMyName

```
<label for="first-name">First name</label>
<input type="text" id="first-name" placeholder="Avi"/>
<br/>
<label for="last-name">Last name</label>
<input type="text" id="last-name" placeholder="Cohen"/>
<br/>
<button id="button1" onclick="sayMyName();" >Click here!</button>
```

- This is another attribute called “onClick”
- It can be added to any HTML element and can contain any valid JavaScript Code
- Usually we will invoke a function
- Let's debug it!

# What is going on? - Code Flow

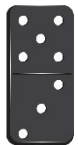
This methodology is called “Event Driven Programming”



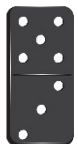
Our program will react to the user's actions



When the user will click the button, our function will be called and executed



The function will then access the fields with id “first-name” and “last-name” and will get their value

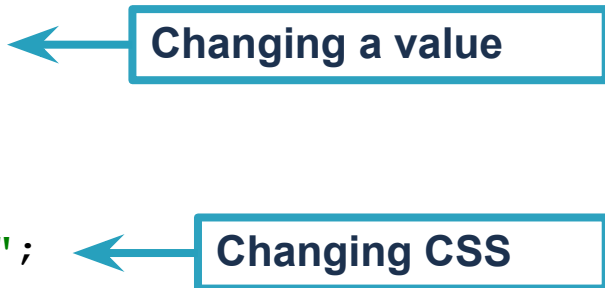


And print the content of the fields to the console

# Change Style

JavaScript can also alter the field's value and style

```
var testMyName = function () {  
    var fname = document.getElementById("first-name");  
  
    if (fname.value === "Dana") {  
        fname.value = "too common";  
        fname.style.color = "red";  
    } else {  
        fname.value = "Nice name";  
        fname.style.color = "green";  
    }  
};
```



```
<button id="button1" onclick="testMyName();" >Click here!</button>
```

# Changing html attributes

We saw how to change the input value

```
var element = document.getElementById("element");  
element.value = "new value";
```

and style:

```
element.style.background = "salmon";
```

We can also change other html attributes:

```
imgElement.src = "http://new-src.com";
```

Or

```
inputElement.type = "checkbox";
```

Or

You get the idea...

# Changing HTML via JS

We can also change the html content of an element!

The innerHTML property sets or returns the HTML content (inner HTML) of an element.

Html content = means everything inside the tag.  
Could be either text or other html elements.

```
document.getElementById("myDiv").innerHTML = "<ul><li>one</li><li>two</li></ul>";
```

# Example – Add 1

Let's create a simple app

Here is our HTML:

```
<label for="number">Put a Number</label>
<input id="number" type="text" />
<button onclick="add1();">Add 1</button>
<div id="result"></div>
```

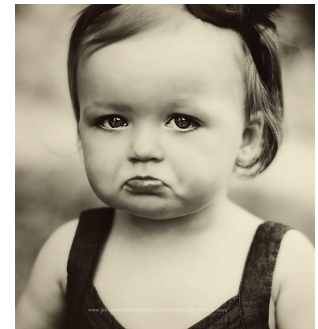
And this is how it looks:

Put a Number

We need to write a function to add 1:

```
function add1() {
    var number = document.getElementById("number").value;
    document.getElementById("result").innerHTML = number + 1;
}
```

**Let's try it**  
**Here is the code!**



# Parsing

The value that we got from the input was a string. But we need a number in order to calculate the result.

- We need to transform input from a string format to a number (in other cases maybe to Boolean...)
  - We can use existing JS functions for that!
- parseInt: string □ whole number
  - parseFloat: string □ decimal number
  - Both are of type number, but parseInt ignores what comes after the decimal point

```
var num1 = parseInt("1234"); //will be 1234
var num2 = parseInt("123.999"); //will be 123
var num3 = parseFloat("123.999"); //will be 123.999
var num4 = parseInt(""); //will be NaN
var num5 = parseInt("One"); //will be NaN
var num6 = parseFloat("One point nine"); //will be NaN
```

# Example – solution

Let's fix the function

```
function add1() {  
  var number = document.getElementById("number").value;  
  var parsedNumber = parseInt(number);  
  document.getElementById("result").innerHTML = parsedNumber+1;  
}
```

Let's Try it again!



# Query functions

- If getting an element by its id is not enough, we can use other means of querying for it.

- `getElementsByTagName("tagName")`
- `getElementsByClassName("myClass")`

- Both return a collection of elements... Why?

Since we can have more than one element with the same class/tag name

- If no such elements exists, we will get an empty collection.

# Query functions Example

```
var elms0 = document.getElementsByClassName('tiny');  
for (var i = 0; i < elms0.length; i++) {  
    var element = elms0[i];  
    console.log(element.textContent);  
}
```

```
<div id="hello"></div>  
<div id="goodbye"></div>  
<div class="hello"></div>  
<div class="goodbye">  
    <span class="tiny">a</span>  
    <span class="tiny">b</span>  
</div>
```

What is going to be the length of elms0? 2

What will this code log to console? a

b

What is the value of elms0[2] ?

undefined

# Questions?

```
console.log("Questions?");
```

# Summary

- You needed to understand:
  - How do we query for elements
  - How to tie an HTML element to an event
  - What event-driven programming is
- You need to remember:
  - What is the wrong way to communicate to the user
  - What NaN is and how to check if a value is nan
- You need to be able to do:
  - Use input types
  - Parse strings to numbers
  - Query for elements
  - Create basic onClick events

# Cheat Sheet

## Input

**Text:** `<input type="text" value="default"/>`

```
    <input type="text" placeholder="email"/>
::-webkit-input-placeholder {
  color: red;
}
```

**With label:** `<label for="is-over-18"></label>`

```
    <input type="checkbox" id="is-over-18"/>
```

**Button:** `<input type="button" value="click me!"/>`  
`<button id="btn-1">Click Here!</button>`

## Get input value:

```
document.getElementById("first-name").value;
```

## Bind events

```
<button id="btn" onclick="func();">Click!</button>
```

## Get html elements / query functions

```
getElementsTagName("tagName");
getElementsByClassName("myClass");
```

} Return collection of elements

## Prompt

```
var userInput = prompt("Your name");
var userInput = confirm('Say yes');
alert("Hello World");
```

## NaN = Not a Number

```
NaN !== NaN
```

## Change style

```
fname.style.color = "green";
```

## Change value

```
fname.value = "Nice name";
```

## Inner HTML

```
document.getElementById("first").innerHTML = "<span>one</span>";
```