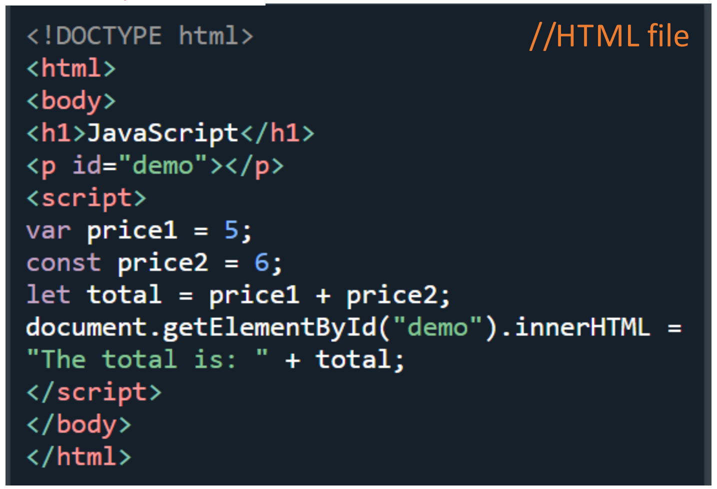
Part IV: JavaScript

JS - Introduction

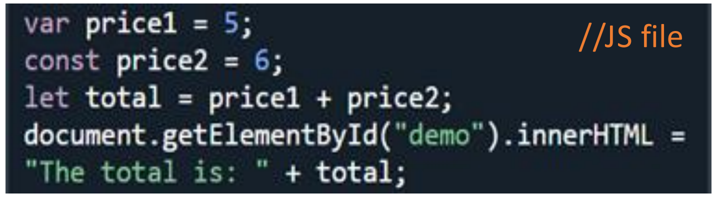
- JavaScript (ECMA-262) 3rd Irreplaceable Web Language
- A web programming language, the most popular, distinct from Java
- Used to provide behavior (dynamic) to web pages
- Interaction, changing: tags, attributes, and CSS style
- Use <script> to insert JS code internally (HTML)
- Use <script src="path_file.js"> to place JS code in an external file

JS - Why ? How ?

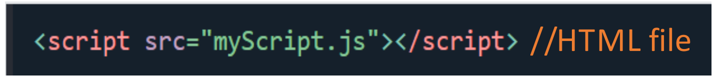
Internal/External



```
<!DOCTYPE html>                          //HTML file
<html>
<body>
<h1>JavaScript</h1>
<p id="demo"></p>
<script>
var price1 = 5;
const price2 = 6;
let total = price1 + price2;
document.getElementById("demo").innerHTML =
"The total is: " + total;
</script>
</body>
</html>
```

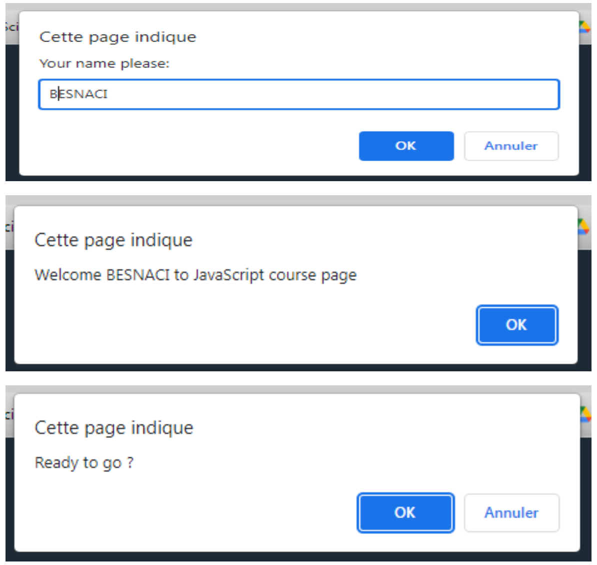


```
var price1 = 5;                          //JS file
const price2 = 6;
let total = price1 + price2;
document.getElementById("demo").innerHTML =
"The total is: " + total;
```



```
<script src="myScript.js"></script> //HTML file
```

Interaction



Cette page indique

Your name please:

BESNACI

OK Annuler

Cette page indique

Welcome BESNACI to JavaScript course page

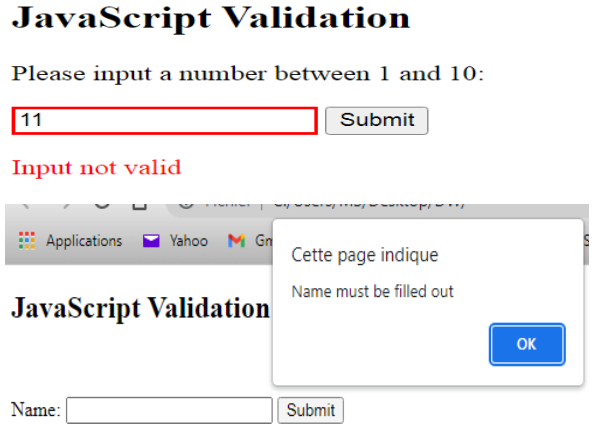
OK

Cette page indique

Ready to go ?

OK Annuler

Control



JavaScript Validation

Please input a number between 1 and 10:

11 Submit

Input not valid

Applications Yahoo Gm

JavaScript Validation

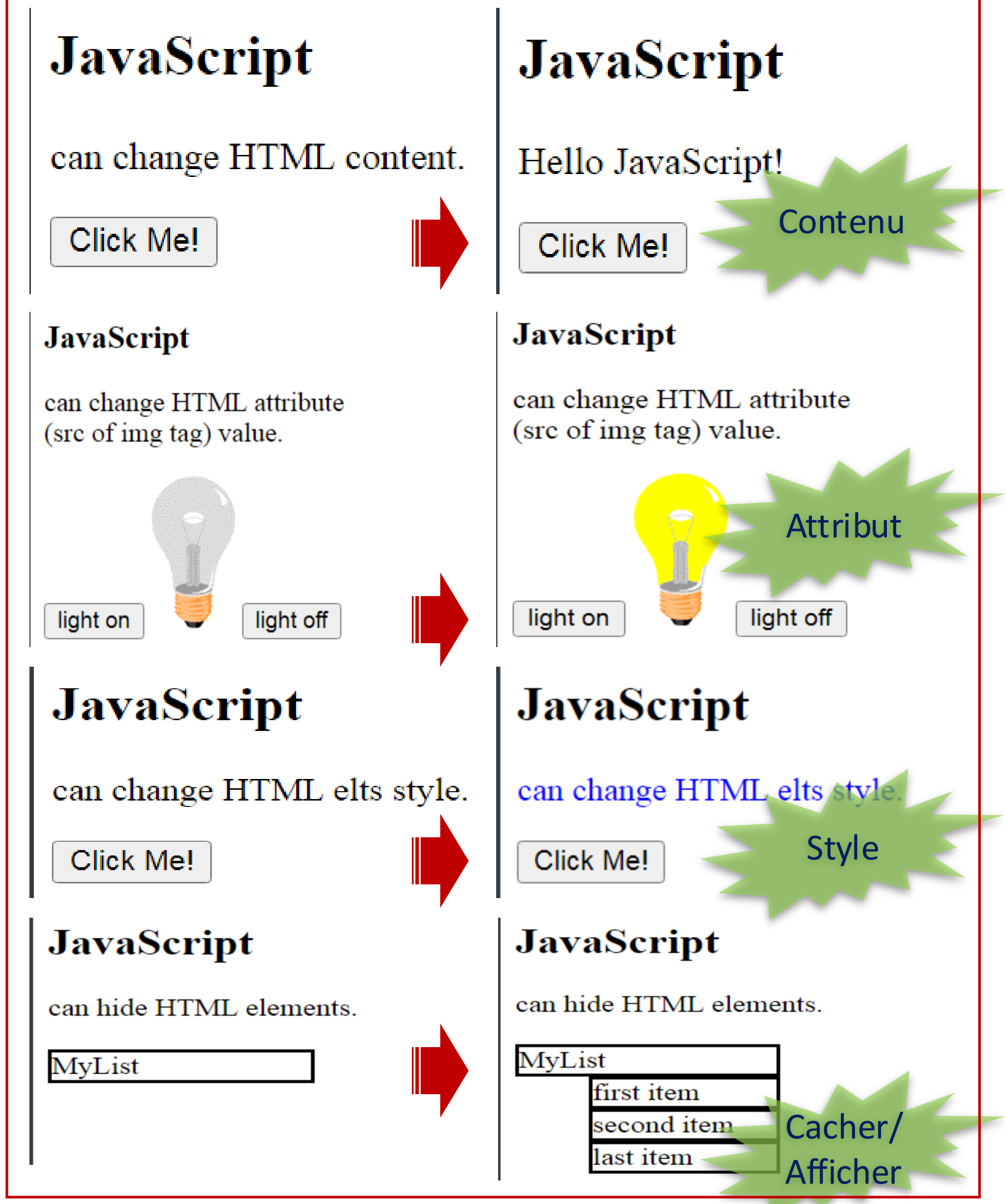
Cette page indique

Name must be filled out

OK

Name: Submit

Modification



JavaScript

can change HTML content.

Click Me!

JavaScript

Hello JavaScript!

Click Me!

JavaScript

can change HTML attribute (src of img tag) value.

light on light off

JavaScript

can change HTML attribute (src of img tag) value.

light on light off

JavaScript

can change HTML elts style.

Click Me!

JavaScript

can change HTML elts style.

Click Me!

JavaScript

can hide HTML elements.

MyList

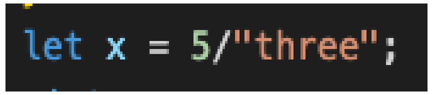
JavaScript

can hide HTML elements.

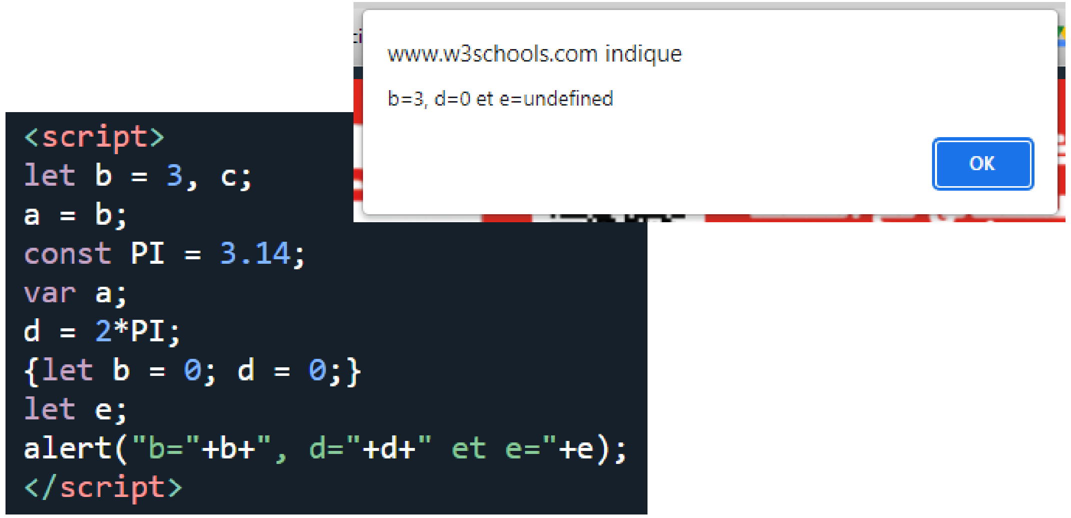
MyList
- first item
- second item
- last item

JS – Variables & types

- Variable Declaration: var, let, and const
  - After declaration, the value = undefined
  - Explicit declaration not mandatory
  - const: defines a fixed value, initialization mandatory
  - let: declare before use, block scope
  - var: declare even after use, global scope
- Variable types: number, string, boolean, array, object, function, ...
  - Implicit and dynamic
  - Use the typeof operator to determine the type of a value
  - NaN indicating a non-numeric quantity
  - isNaN() returns true if NaN and false otherwise

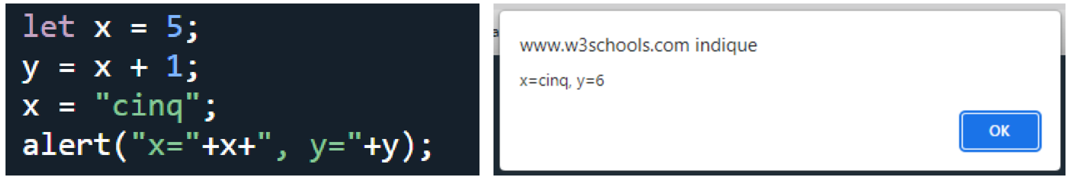


```
let x = 5/"three";
```



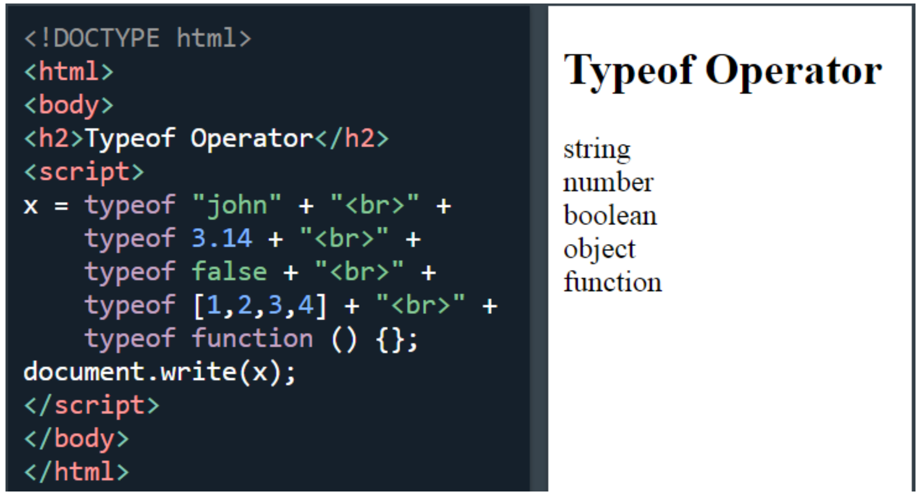
```
<script>
let b = 3, c;
a = b;
const PI = 3.14;
var a;
d = 2*PI;
{let b = 0; d = 0;}
let e;
alert("b="+b+", d="+d+" et e="+e);
</script>
```

```
www.w3schools.com indique
b=3, d=0 et e=undefined
```



```
let x = 5;
y = x + 1;
x = "cinq";
alert("x="+x+", y="+y);
```

```
www.w3schools.com indique
x=cinq, y=6
```

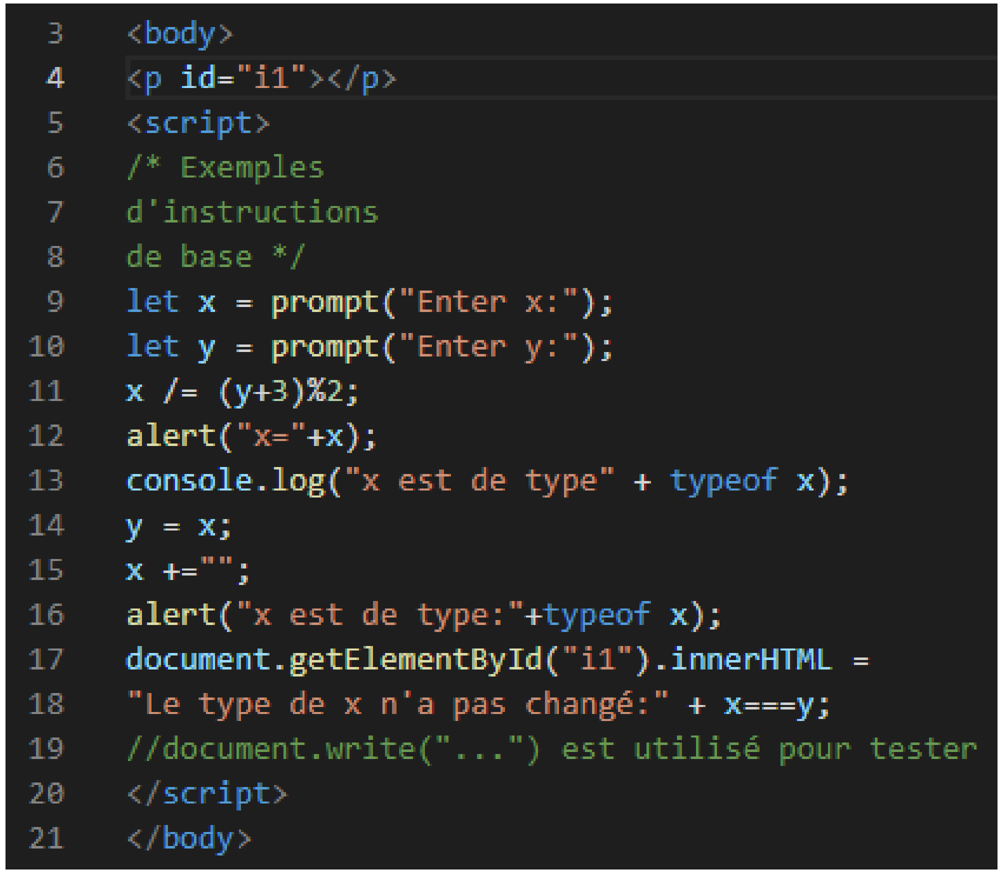


```
<!DOCTYPE html>
<html>
<body>
<h2>Typeof Operator</h2>
<script>
x = typeof "john" + "<br>" +
    typeof 3.14 + "<br>" +
    typeof false + "<br>" +
    typeof [1,2,3,4] + "<br>" +
    typeof function () {};
document.write(x);
</script>
</body>
</html>
```

```
Typeof Operator

string
number
boolean
object
function
```

JS – basic instructions

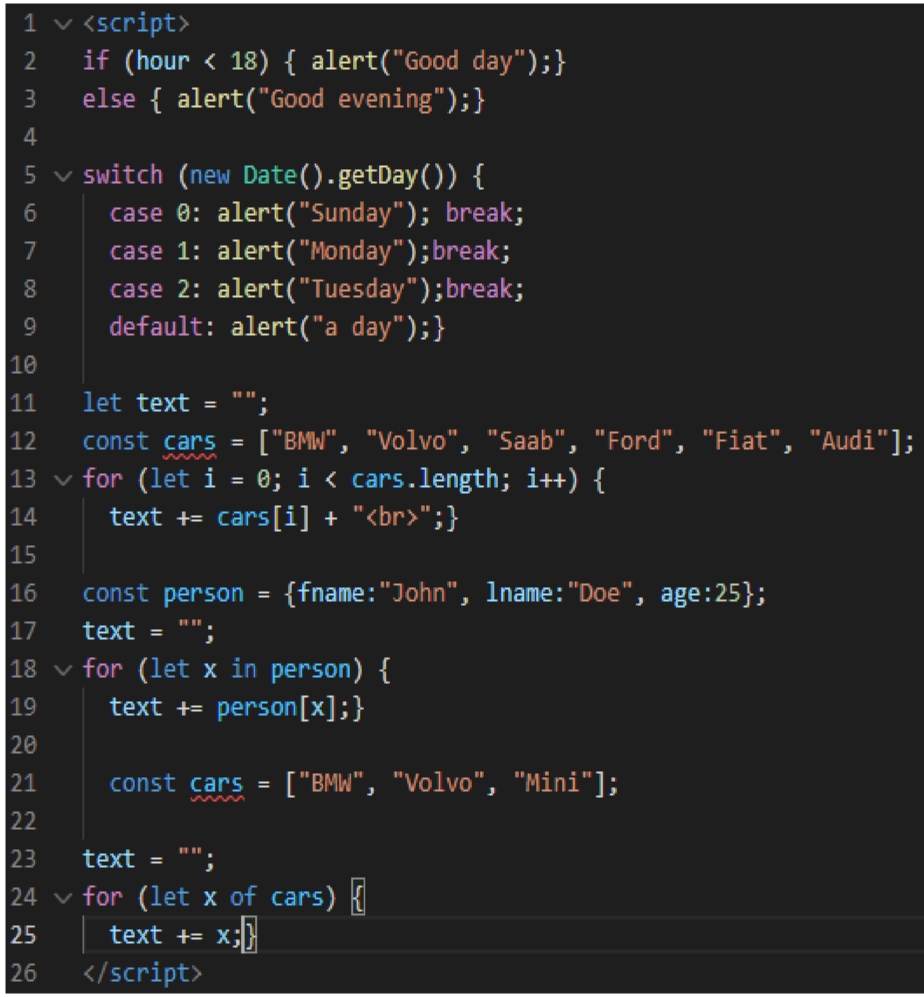
- Expressions: combination of operators (+, -, &&, !, ==, ===, !=, >=, ?, **typeof**, ...) and operands (literals, variables, functions) - classic syntax
- Assignment: =, +=, *=, %=, ...
- I/O: dialog boxes, outputs on the HTML document, on the browser console
  - Dialog boxes: **alert**(), **confirm**(), **prompt**()
  - On HTML: **document.write**(), **innerHTML** attribute (DOM functions)
  - On console: **console.log**()
- Comments: // single line, /* multiline */
- JavaScript is case-sensitive (identifiers)



```
<body>
<p id="i1"></p>
<script>
/* Exemples
d'instructions
de base */
let x = prompt("Enter x:");
let y = prompt("Enter y:");
x /= (y+3)%2;
alert("x="+x);
console.log("x est de type" + typeof x);
y = x;
x +="";
alert("x est de type:"+typeof x);
document.getElementById("i1").innerHTML =
"Le type de x n'a pas changé:" + x===y;
//document.write("...") est utilisé pour tester
</script>
</body>
```

JS – control instructions

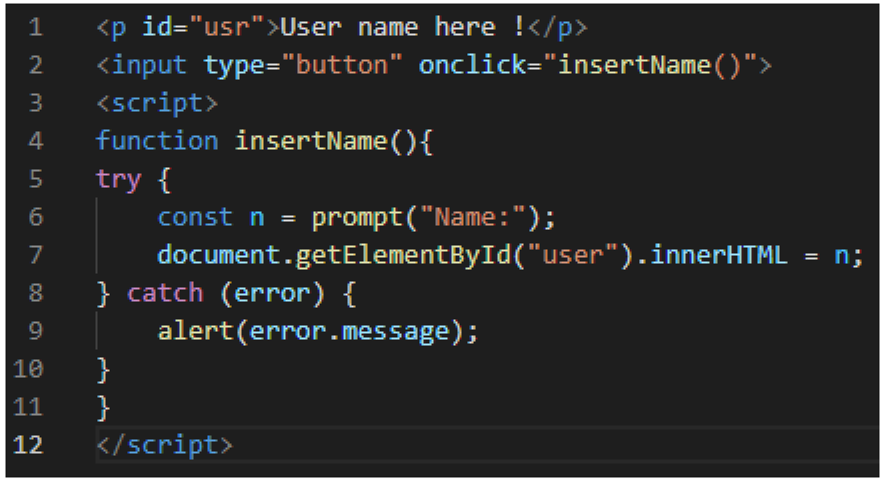
- Conditionnel
  - **if** (Cond) Inst / **if** (Cond) Inst **else** Inst → classic
  - **switch** (Exp) {**case** Val: Inst break; … **default**: Inst} → comparison with ===
- Repeatition
  - **for** (Init; Cond; Incr) Inst → classic
  - **for** (Var **in** Obj) Inst → iterate over an object's properties
  - **for** (Var **of** Itér) Inst → iterate over the values of an iterable
  - **while** (Cond) Inst → classic
  - **do** Inst **while** (Cond) → classic



```
<script>
if (hour < 18) { alert("Good day");}
else { alert("Good evening");}

switch (new Date().getDay()) {
  case 0: alert("Sunday"); break;
  case 1: alert("Monday");break;
  case 2: alert("Tuesday");break;
  default: alert("a day");}

let text = "";
const cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];
for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";}

const person = {fname:"John", lname:"Doe", age:25};
text = "";
for (let x in person) {
  text += person[x];}

  const cars = ["BMW", "Volvo", "Mini"];

text = "";
for (let x of cars) {
  text += x;}
</script>
```

JS – functions, events, errors

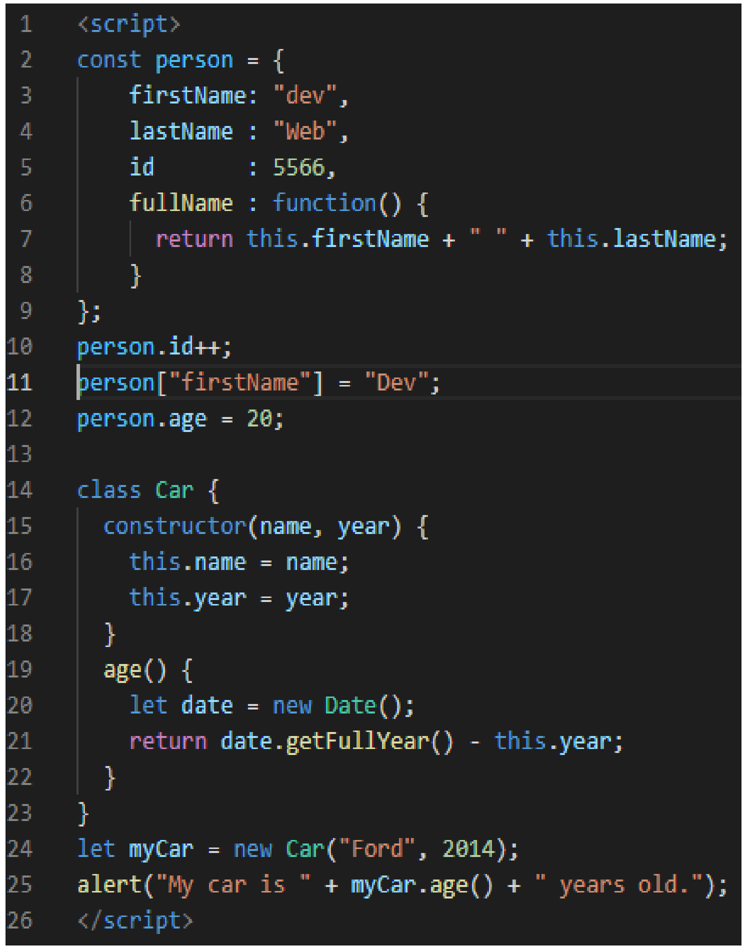
- Functions
  - **function** Name(P1, ...) {code}
  - Function with **return** to send back values
  - Call **without parentheses** to return the code
  - **Typeof** returns **function** for functions
- Events
  - Controlling events triggered by the **browser** or the **user**
    - E.g., Page loading/closing, ...
    - E.g., Data filling, triggering actions, ...
  - HTML provides attributes to manage them
    - E.g., onload, onchange, onclick, onmouseover, onkeypress, ...
- Errors
  - **try** {code} **catch**(err) {code}
  - Handling code that may cause an error (try)
  - Handling the error by executing certain code (catch)



```
<p id="usr">User name here !</p>
<input type="button" onclick="insertName()">
<script>
function insertName(){
try {
    const n = prompt("Name:");
    document.getElementById("user").innerHTML = n;
} catch (error) {
    alert(error.message);
}
}
</script>
```

JS – objects & classes

- Objects
  - **const** obj = {prop1:val1, …propi: **function**(…){…}, … }
    - Ex: const car = {type:"Fiat", model:"500", color:"white"};
  - obj.prop = val / obj["prop"] = val / obj.propi(…)
    - Ex: alert(car.type); car["model"] = "100"; car.move();
- Classes
  - **class** name { constructor(param) { Init } m1(…) { … } … }
  - **constructor**: object creation and property initialization
  - **const** obj = **new** name(…); class object creation
  - **Implicit** properties in the constructor
  - obj **instanceof** type: true if obj is an instance of type

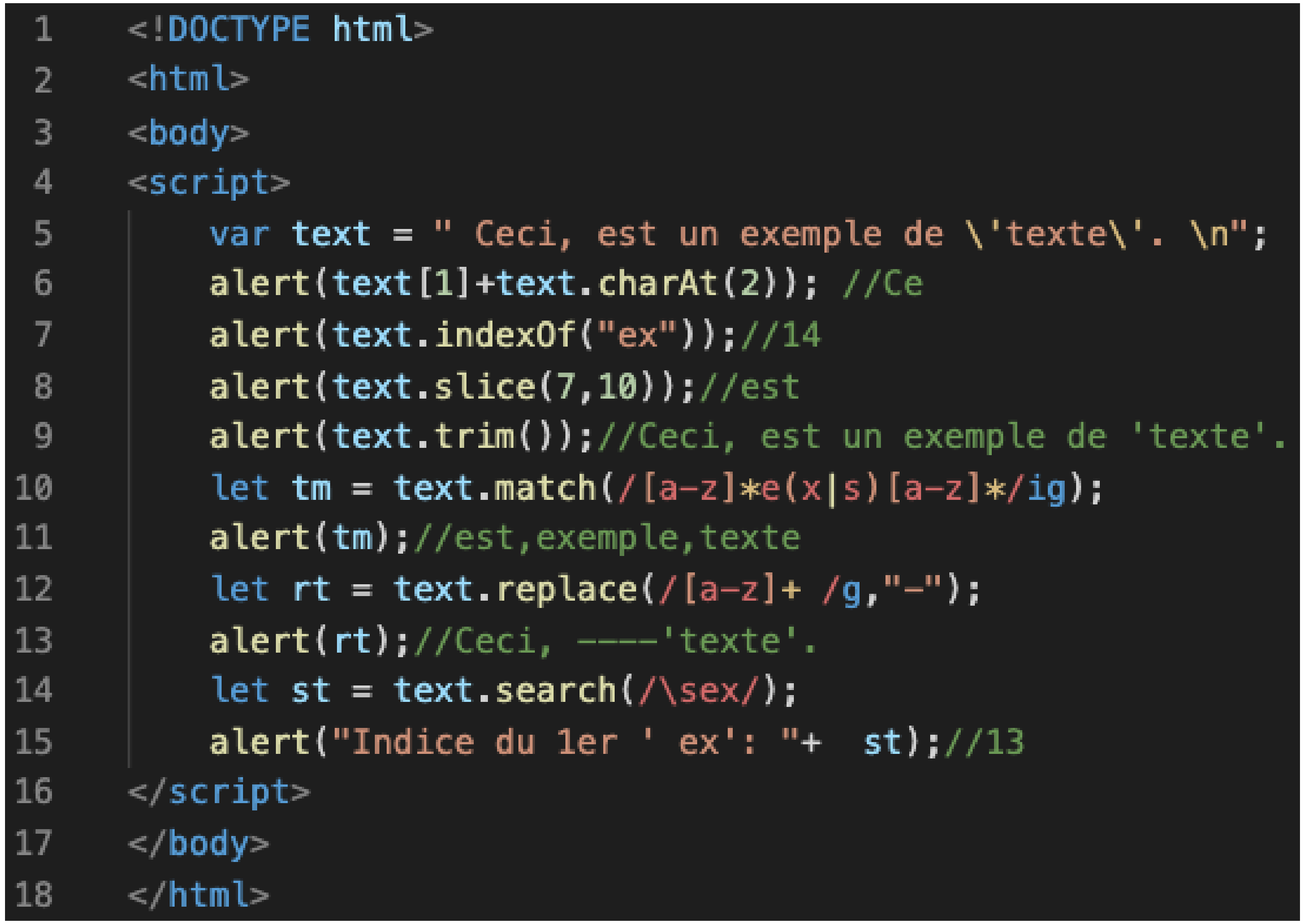


```
<script>
const person = {
    firstName: "dev",
    lastName : "Web",
    id       : 5566,
    fullName : function() {
      return this.firstName + " " + this.lastName;
    }
};
person.id++;
person["firstName"] = "Dev";
person.age = 20;

class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  age() {
    let date = new Date();
    return date.getFullYear() - this.year;
  }
}
let myCar = new Car("Ford", 2014);
alert("My car is " + myCar.age() + " years old.");
</script>
```

JS – String object

- Strings: used to store and manipulate text (' ' / " ")
- Special characters: \", \', \\, \n, \t, ...
- Regular expressions (regex): string representing a search pattern
  - **Syntax**: /pattern/modifiers → e.g., /algeria(-|_)\d*/i e.g., /[0-5]+\s-\s(da|dt)/g
- Length: str.length
- Methods:
  - **charAt**(index): character at index "index"
  - **indexOf**(str): index of the first occurrence of a string "str"
  - **match**(rex): array of occurrences of a string/regex "rex" (or null)
  - **replace**(rex,str): replaces occurrences of a string/regex "rex" with another string "str"
  - **search**(rex): index of the first occurrence (or -1) of a string/regex "rex"
  - **slice**(index,length): part of the string starting from index "index" of length "length"
  - **split**(sep): array of substrings according to the separator "sep"
  - **trim**(): string without space characters, tab, ... at the ends

JS – String object

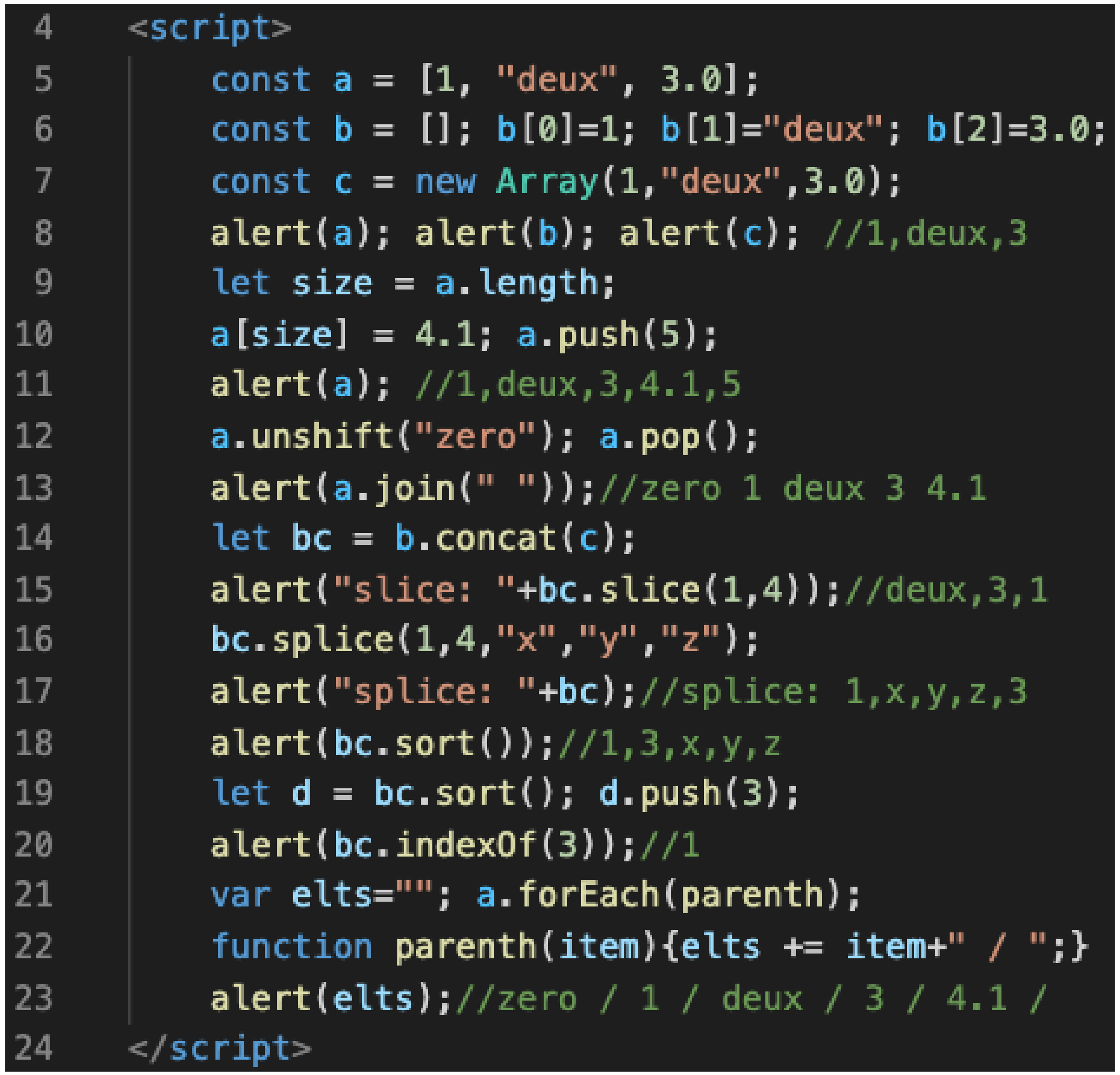


```
<!DOCTYPE html>
<html>
<body>
<script>
    var text = " Ceci, est un exemple de \'texte\'. \n";
    alert(text[1]+text.charAt(2)); //Ce
    alert(text.indexOf("ex"));//14
    alert(text.slice(7,10));//est
    alert(text.trim());//Ceci, est un exemple de 'texte'.
    let tm = text.match(/[a-z]*e(x|s)[a-z]*/ig);
    alert(tm);//est,exemple,texte
    let rt = text.replace(/[a-z]+ /g,"-");
    alert(rt);//Ceci, ----'texte'.
    let st = text.search(/\sex/);
    alert("Indice du 1er ' ex': "+  st);//13
</script>
</body>
</html>
```

JS – Array object

- Creation: const t=[1, 'two', 3.0] / const t=[]; t[0]=1;t[1]='two';t[2]=3.0; / const t=new Array(1,'two',3.0)
- Size (number of elements): t.length
- Adding elements: t[t.length]=val / t.push(val)
- Verification: Array.isArray(t)
- Methods:
  - **t.push(elt)/t.pop(), t.unshift(elt)/t.shift()**: push/pop at beginning/end
  - **t.join**(sep): list of elements separated by sep
  - **t1.concat**(t2, …): concatenation of multiple arrays
  - **t.slice**(i,n): portion starting from index i with n-1 elements
  - **t.splice**(i,n,e1,…): replace n elements from index i with elements ei (returns replaced elements and changes t)
  - **t.sort()/t.reverse()**: ascending/descending alphanumeric sort
  - **t.indexOf**(e): index of the first occurrence of element e
  - **t.forEach**(f): call function f for each element

JS – Array object



```
<script>
    const a = [1, "deux", 3.0];
    const b = []; b[0]=1; b[1]="deux"; b[2]=3.0;
    const c = new Array(1,"deux",3.0);
    alert(a); alert(b); alert(c); //1,deux,3
    let size = a.length;
    a[size] = 4.1; a.push(5);
    alert(a); //1,deux,3,4.1,5
    a.unshift("zero"); a.pop();
    alert(a.join(" "));//zero 1 deux 3 4.1
    let bc = b.concat(c);
    alert("slice: "+bc.slice(1,4));//deux,3,1
    bc.splice(1,4,"x","y","z");
    alert("splice: "+bc);//splice: 1,x,y,z,3
    alert(bc.sort());//1,3,x,y,z
    let d = bc.sort(); d.push(3);
    alert(bc.indexOf(3));//1
    var elts=""; a.forEach(parenth);
    function parenth(item){elts += item+" / ";}
    alert(elts);//zero / 1 / deux / 3 / 4.1 /
</script>
```

JS – Date object

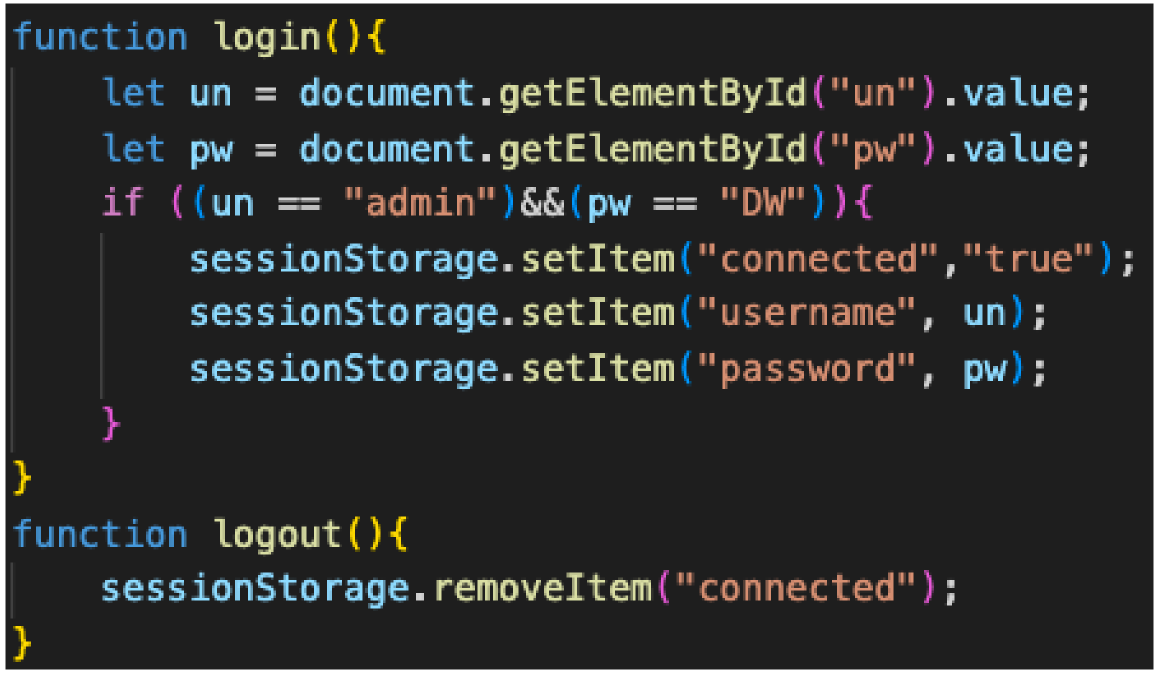
- Creation: const d=new Date(); / const d=new Date(y,m,d,h,mt,s,ms); / const d=new Date(str); / const d=new Date(ms_1970)
- Formats (strings):
  - **ISO**: YYYY-MM-DD (2022-03-31), YYYY-MM (2022-03), YYYY (2022), YYYY-MM-DDTHH:MM:SSZ (2022-03-31T21:05:23Z)
  - **Short**: MM/DD/YYYY (06/15/2002)
  - **Long**: MMM DD YYYY (Apr 15 2000), MMM… DD YYYY (April 10 2015)
- Display: use d.toDateString() for a nicer display
- Methods:
  - **Date.parse**(str): milliseconds of a valid date e.g., Date.parse("March 21, 2012") → 1332284400000
  - **d.getFullYear(), d.getMonth(), d.getDate(), d.getHours(), …, d.getTime(), d.getDay(), Date.now()**: respectively give the year, month, day, hour, …, time in milliseconds, day of the week, and current time
  - **d.setFullYear()**, …: respectively set the year, …

JS – Math object

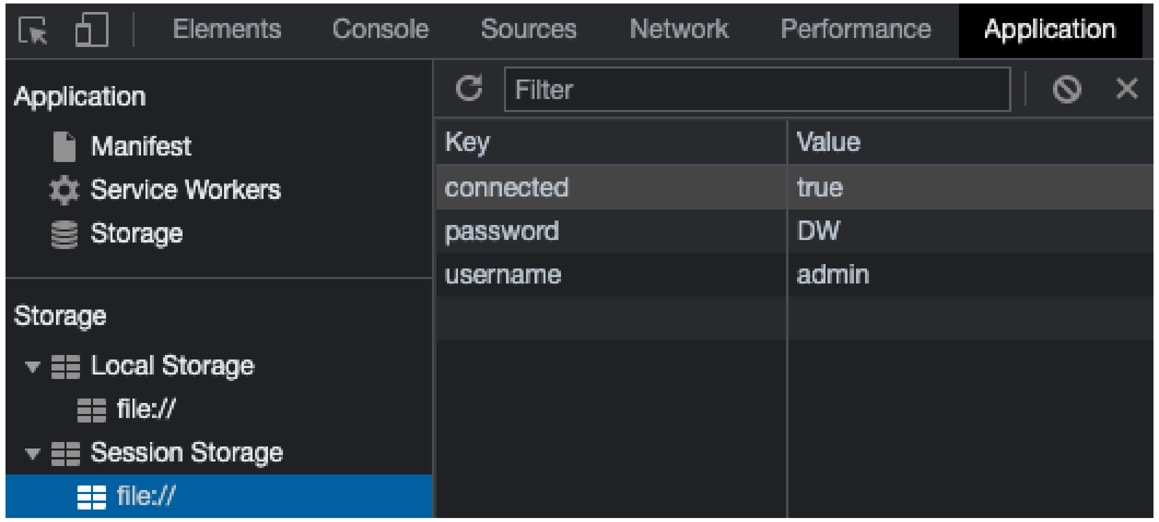
- Math is a static object, **Math.PI, Math.E, Math.LN2**, … are constants
- Methods
  - **Math.round**(x): x rounded to its nearest integer
  - **Math.trunc**(x): integer part of x
  - **Math.sign**(x): returns 1 if x is positive, -1 if negative, or 0 if zero
  - **Math.power**(x,y): x raised to the power y
  - **Math.sqrt**(x): square root of x
  - **Math.abs**(x): absolute value of x
  - **Math.sin/cos**(x): sine/cosine of x
  - **Math.min/max**(x1,x2, …): minimum/maximum of a list of arguments
  - **Math.random**(): random number between 0 and 1
  - **Math.log**(x): natural logarithm of x

JS – Data Storage objects

- To create and manipulate data locally in the browser for a specific website
- Data format: ”key-value” pairs
- Duration: **permanent** (localStorage object) or **temporary** (sessionStorage object)
    - setItem("key","value") → insertion
    - getItem("key") → retrieval
    - removeItem("key") → deletion
    - clear() → deletion of all data



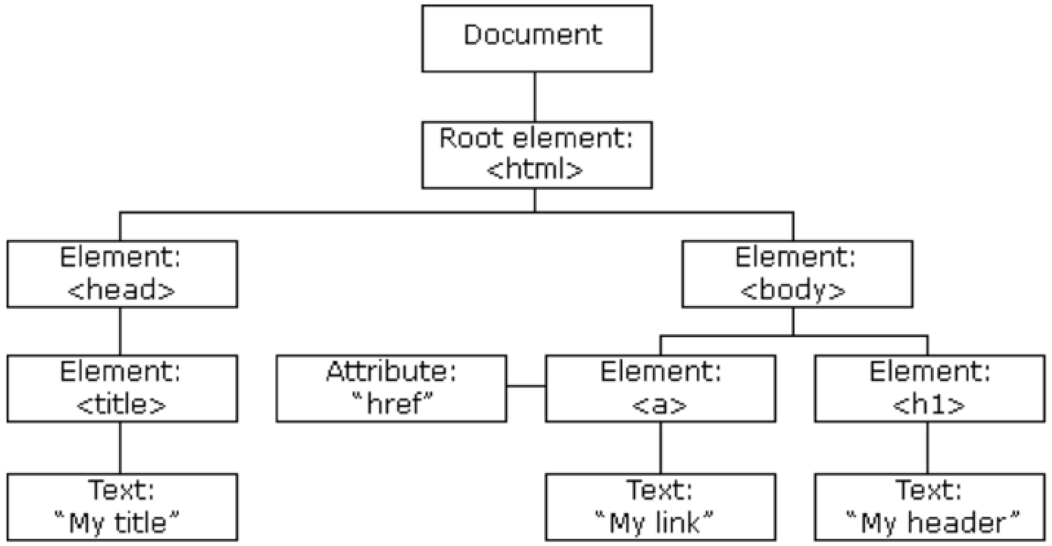
```
function login(){
    let un = document.getElementById("un").value;
    let pw = document.getElementById("pw").value;
    if ((un == "admin")&&(pw == "DW")){
        sessionStorage.setItem("connected","true");
        sessionStorage.setItem("username", un);
        sessionStorage.setItem("password", pw);
    }
}
function logout(){
    sessionStorage.removeItem("connected");
}
```

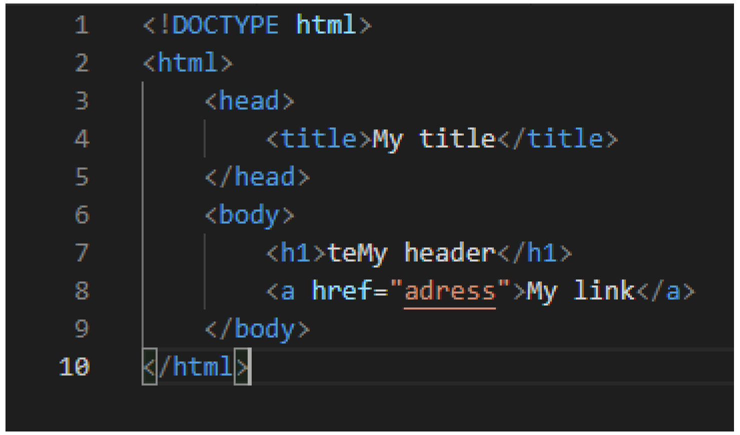


Key	Value
connected	true
password	DW
username	admin

JS – HTML DOM

- DOM (Document Object Model): a **standard** object model for HTML documents, defining:
  - HTML elements as objects
  - Properties of all HTML elements
  - Methods for accessing all HTML elements
  - Events for all HTML elements
- HTML DOM is a **standard** for how to get, modify, add, or remove HTML elements.

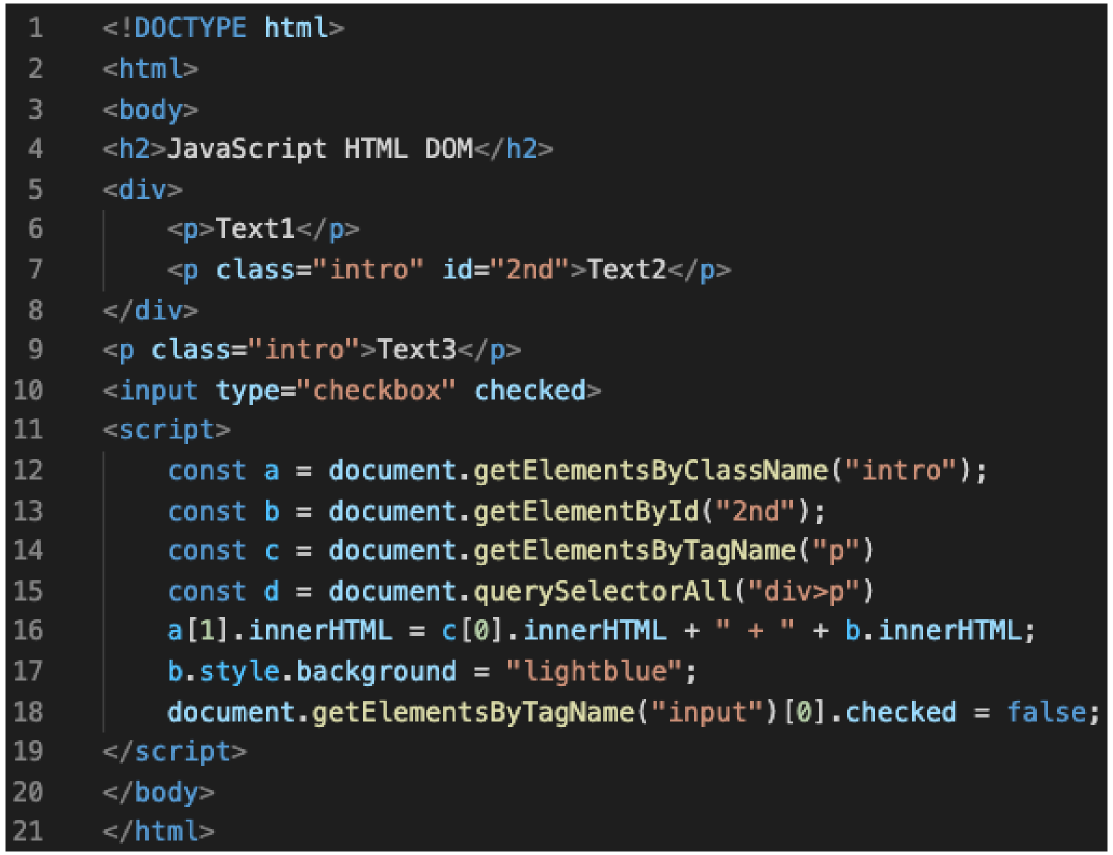




```html
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <h1>teMy header</h1>
        <a href="adress">My link</a>
    </body>
</html>
```

JS – DOM HTML

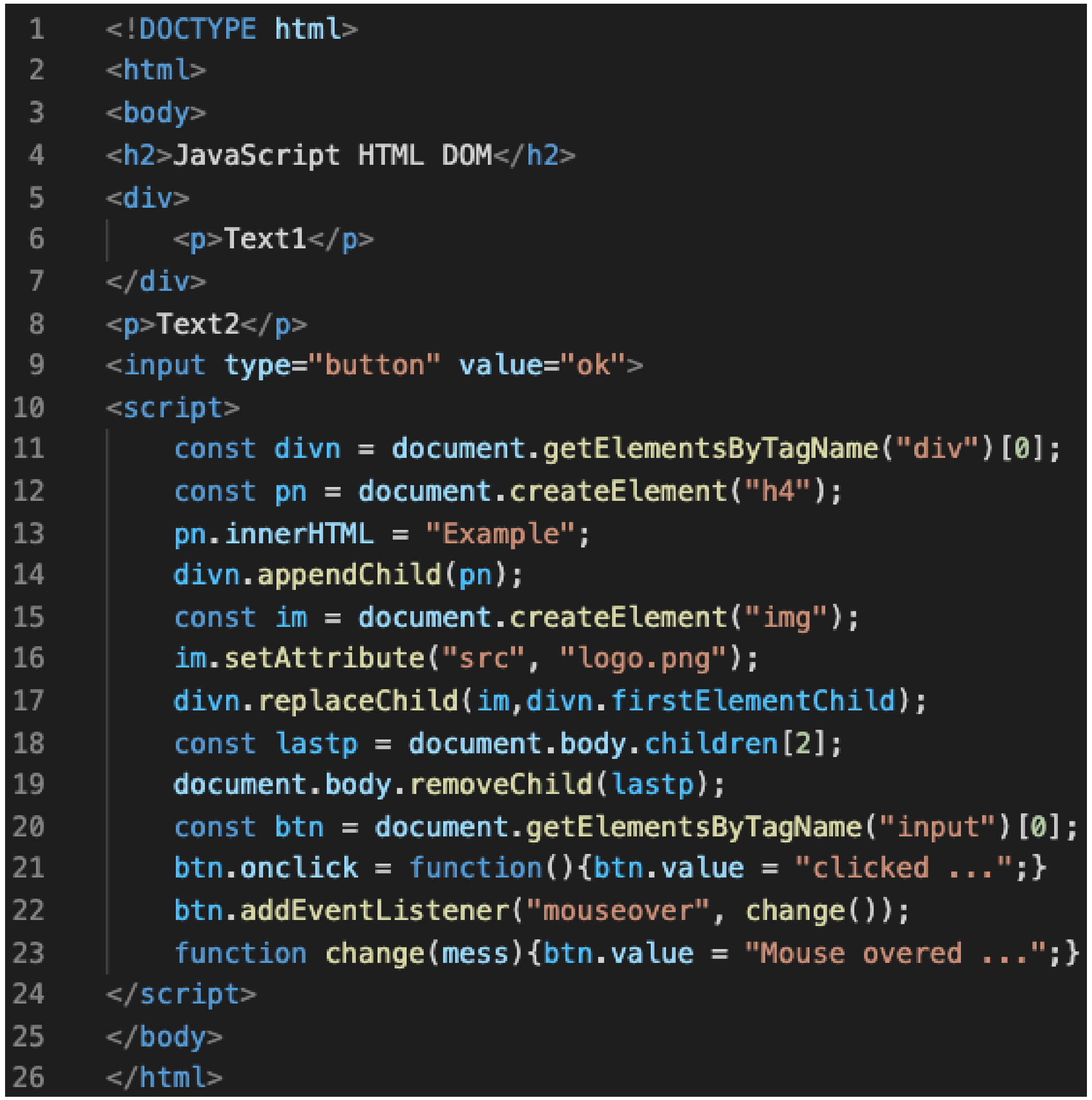
- **Manipulating HTML elements** involves manipulating DOM objects through their **properties** (e.g., innerHTML) and **methods** (e.g., getElementById).
- Search for elements
  - ducument.getElementById(id)
  - ducument.getElementsByTagName(tag)
  - ducument.getElementsByClassName(class)
  - document.querySelector(selector)
  - document.querySelectorAll(selector)
- Update elements
  - e.innerHTML = val
  - e.attribute_name = val
  - e.style.property_name = val
  - e.setAttribute(attribute_name, val)



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript HTML DOM</h2>
<div>
    <p>Text1</p>
    <p class="intro" id="2nd">Text2</p>
</div>
<p class="intro">Text3</p>
<input type="checkbox" checked>
<script>
    const a = document.getElementsByClassName("intro");
    const b = document.getElementById("2nd");
    const c = document.getElementsByTagName("p")
    const d = document.querySelectorAll("div>p")
    a[1].innerHTML = c[0].innerHTML + " + " + b.innerHTML;
    b.style.background = "lightblue";
    document.getElementsByTagName("input")[0].checked = false;
</script>
</body>
</html>
```

JS – DOM HTML

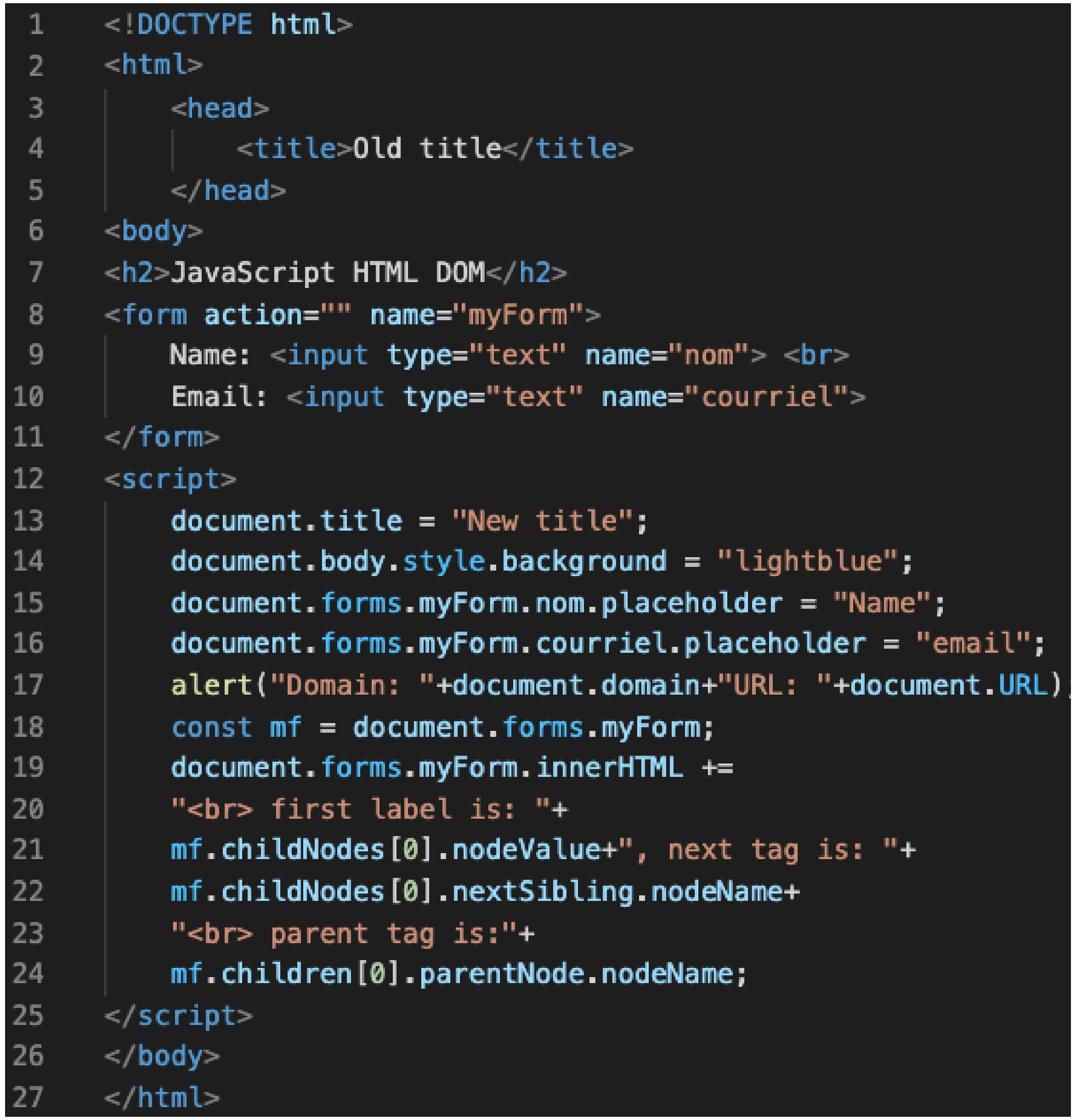
- Adding/removing elements
  - document.createElement(e)
  - document.removeChild(e)
  - document.appendChild(e)
  - document.replaceChild(new,old)
- Add event handler
  - e.onclick = function(){code}
  - *e*.addEventListener(*event, function*)



```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript HTML DOM</h2>
<div>
    <p>Text1</p>
</div>
<p>Text2</p>
<input type="button" value="ok">
<script>
    const divn = document.getElementsByTagName("div")[0];
    const pn = document.createElement("h4");
    pn.innerHTML = "Example";
    divn.appendChild(pn);
    const im = document.createElement("img");
    im.setAttribute("src", "logo.png");
    divn.replaceChild(im,divn.firstElementChild);
    const lastp = document.body.children[2];
    document.body.removeChild(lastp);
    const btn = document.getElementsByTagName("input")[0];
    btn.onclick = function(){btn.value = "clicked ...";}
    btn.addEventListener("mouseover", change());
    function change(mess){btn.value = "Mouse overed ...";}
</script>
</body>
</html>
```

JS – DOM HTML

- Specific HTML elements
  - document.documentElement, document.body, document.head, document.title
  - document.forms, document.images
  - document.URL, document.domain
- Navigate between elements
  - e.parentNode, e.childNodes[n°], e.firstChild, e.lastChild, e.nextSibling, e.previousSibling
  - e.nodeName, e.nodeValue, e.nodeType



```
<!DOCTYPE html>
<html>
    <head>
        <title>Old title</title>
    </head>
<body>
<h2>JavaScript HTML DOM</h2>
<form action="" name="myForm">
    Name: <input type="text" name="nom"> <br>
    Email: <input type="text" name="courriel">
</form>
<script>
    document.title = "New title";
    document.body.style.background = "lightblue";
    document.forms.myForm.nom.placeholder = "Name";
    document.forms.myForm.courriel.placeholder = "email";
    alert("Domain: "+document.domain+"URL: "+document.URL)
    const mf = document.forms.myForm;
    document.forms.myForm.innerHTML +=
    "<br> first label is: "+
    mf.childNodes[0].nodeValue+", next tag is: "+
    mf.childNodes[0].nextSibling.nodeName+
    "<br> parent tag is:"+
    mf.children[0].parentNode.nodeName;
</script>
</body>
</html>
```