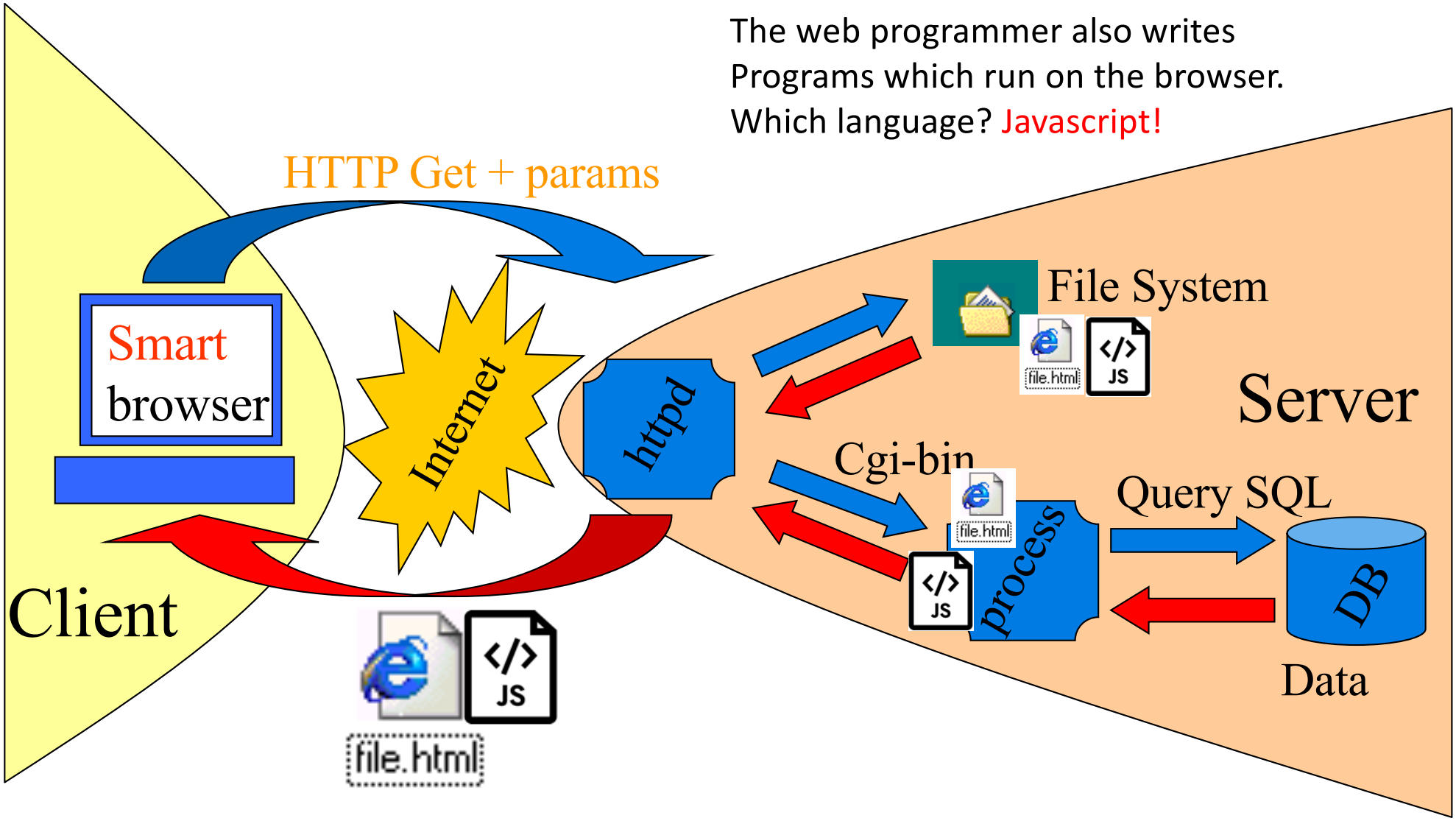
Javascript and the DOM

The web architecture with smart browser

The web programmer also writes
Programs which run on the browser.
Which language? Javascript!



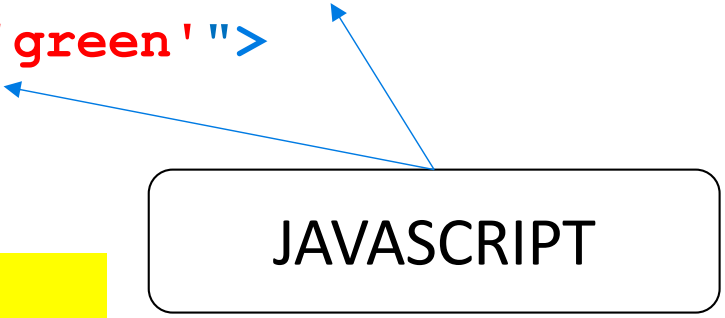
Evolution 3: execute code also on client! (How ?)

Javascript and the DOM

1- Adding dynamic behaviour to HTML

Example 1: onmouseover, onmouseout

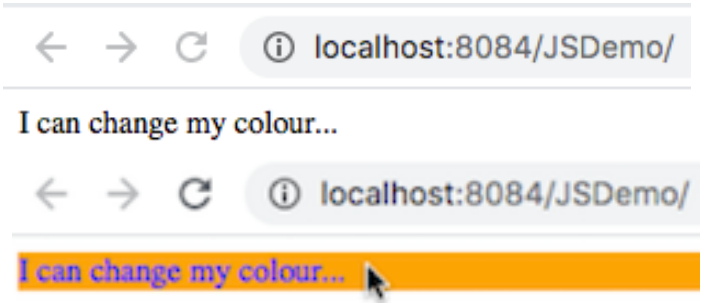
```
<!DOCTYPE html>
<html>
    <head>
        <title>Dynamic behaviour</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
    </head>
    <body>
        <div onmouseover="this.style.color = 'red'"
  onmouseout="this.style.color = 'green'">
  I can change my colour!</div>
    </body>
</html>
```

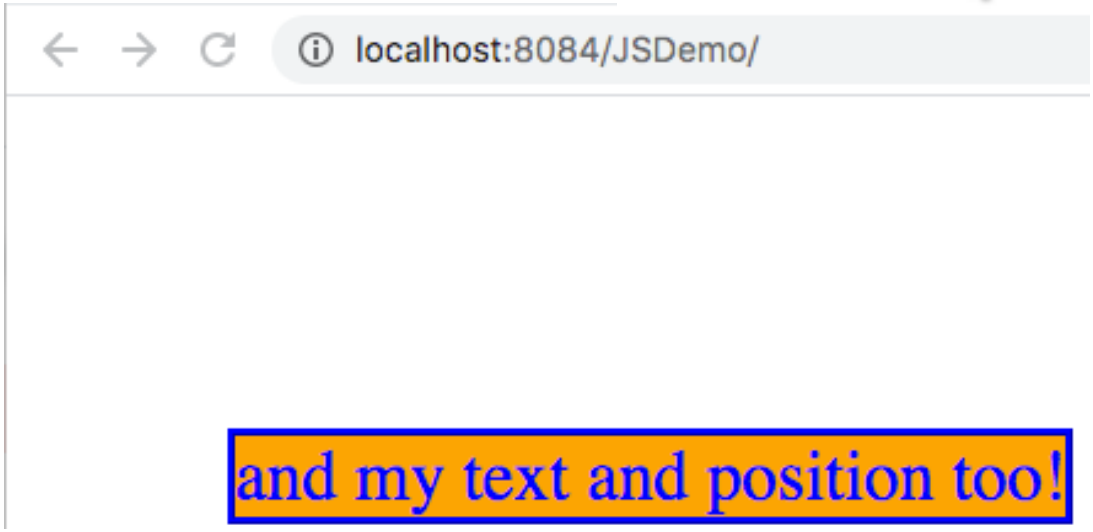


The dynamic behaviour is
on the client side!
(The file can be loaded locally)

Example 2: onmouseover, onmouseout

```
<body>
  <div
    onmouseover="this.style.background='orange';
      this.style.color = 'blue';"
    onmouseout="
      this.innerText='and my text and position too!';
      this.style.position='absolute';
      this.style.left='100px';
      this.style.top='150px';
      this.style.borderStyle='ridge';
      this.style.borderColor='blue';
      this.style.fontSize='24pt';">
   I can change my colour...
  </div>
</body >
```





JavaScript is event-based

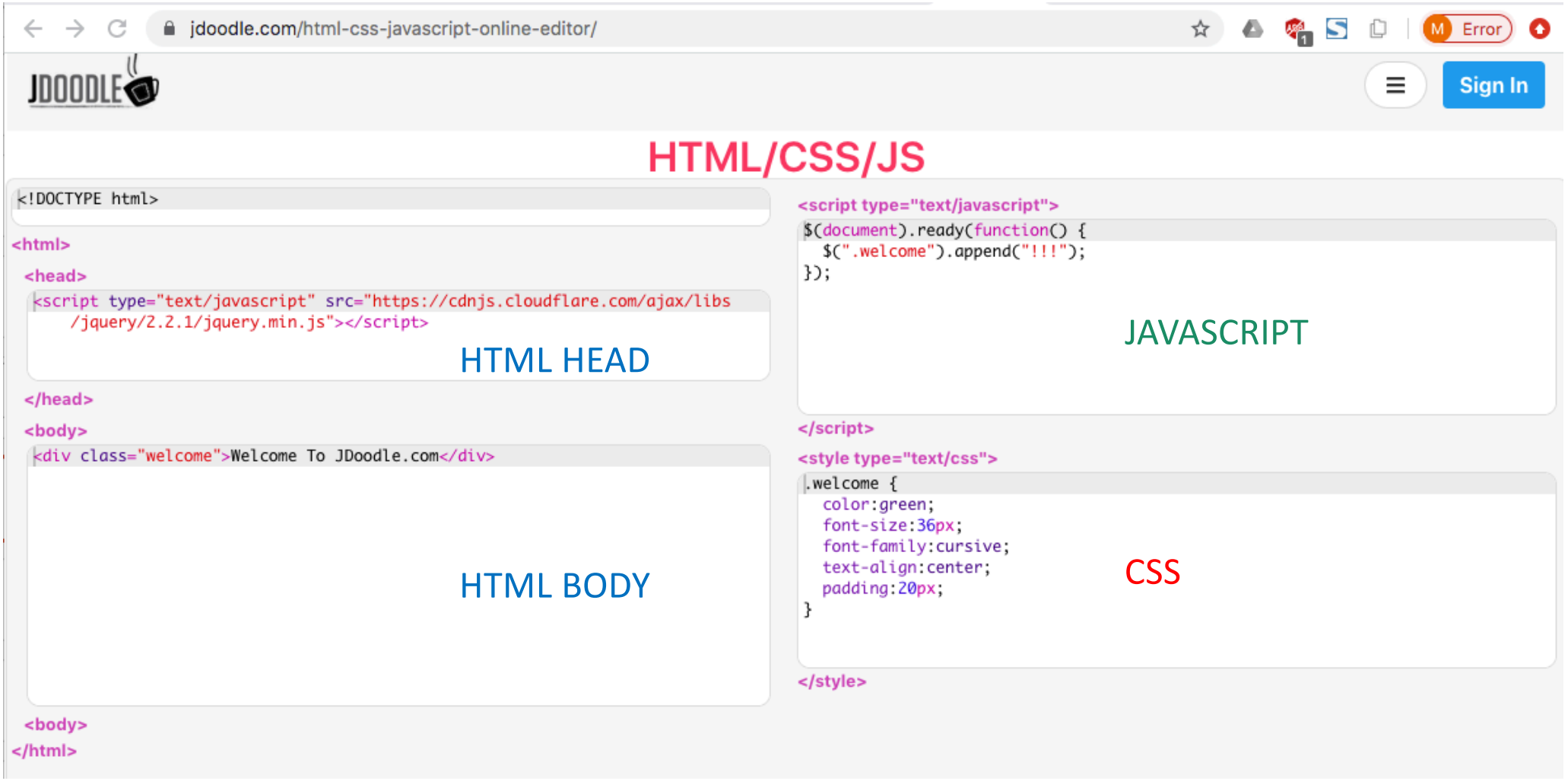
UiEvents:

These event objects iherits the properties of the UiEvent:

- The FocusEvent
- The InputEvent
- The KeyboardEvent
- The MouseEvent
- The TouchEvent
- The WheelEvent

See https://www.w3schools.com/jsref/obj_uievent.asp

Test and Gym



https://www.jdoodle.com/html-css-javascript-online-editor/

Javascript and the DOM

2- Introduction to the language

JavaScript History

- JavaScript was born as Mocha, then "**LiveScript**" at the beginning of the 94's.
- Name changed into JavaScript (name owned by Netscape)
- Microsoft responds with **Vbscript**
- Microsoft introduces **JScript** (dialect of Javascript)
- A standard is defined: ECMAScript (ECMA-262, ISO-16262)
- Jscript survives (as ECMAScript incarnation till 2009, then as Chakra till 2015)
- Another incarnation of ECMAScript is **ActionScript** (Adobe, for Flash)

JavaScript: History

Name	Edition	Date published			
Mocha		May-95			
LiveScript		Sep-95			
JavaScript		Dec-95			
			Jscript	Aug-96	Microsoft
ECMAScript	1	Jun-97			
ECMAScript	2	Jun-98	ActionScript	1998	Macromedia/Adobe
ECMAScript	3	Dec-99			
ECMAScript	4	Abandoned			
ECMAScript	5	Dec-09			
ECMAScript	5,1	Jun-11			
ECMAScript 2015 (ES2015)	6	Jun-15			
ECMAScript 2016 (ES2016)	7	Jun-16			
ECMAScript 2017 (ES2017)	8	Jun-17			
ECMAScript 2018 (ES2018)	9	Jun-18			
ECMAScript 2019 (ES2019)	10	Jun-19			

ECMAScript Engine

An ECMAScript engine is a program that executes source code written in a version of the ECMAScript language standard

Examples:

- **V8 (Chrome, NodeJS, Opera)**
- **SpiderMonkey (Mozilla)**
- **Chakra (Microsoft)**
- **JavaScriptCore(Apple)**
- **Nashorn (Oracle – JDK)**

See https://en.wikipedia.org/wiki/List_of_ECMAScript_engines

JavaScript Myths

- JavaScript is NOT simple
  - **Simple tasks are indeed simple**
- JavaScript is NOT Java

	Java	JavaScript
Browser Control	NO	YES
Networking	YES	Partial
Graphics	YES	Partial

JavaScript is…

- Scripted (not compiled)
- Powerful
- Cross-Platform
- Client and Server

- Object-based
  (and partially oriented)
- Event-driven

JavaScript allows…

- Dynamic Web Sites
- Dynamic HTML (DHTML)
- Interactive Pages/Forms
- Server-Side Functionality
- Application Development

JavaScript and HTML

- Between `<script>` and `</script>` tags
- In a `<script src="url"></script>` tag
- Between `<server>` and `</server>` tags
- In an event handler:

```
<input type="button" value="Click me"
                onClick="js code">
<div onmouseover="this.style.color =
  'red'" onmouseout="this.style.color =
  'green'”>
```

Base

- Syntax is C-like (C++-like, Java-like)

  case-sensitive,

  statements end with (optional) semicolon ;

  //comment /*comment*/

  operators (=,*,+,++,+=,!=,==,&&,…)

Statements

- `if (expression) {statements} else {statements}`
- `switch (expression) {`
  `case value: statements; break;`
  `…`
  `default: statements; break;`
  `}`
- `while (expression) {statements}`
- `do (expression) while {statements}`
- `for (initialize ; test ; increment) {statements}`
  `for (a in s) {statements}`

Data types

- Primitive data types
  number, string, boolean, undefined

- Complex data types
  object, function (more later!)

- Loosely, dynamic typed variables (Basic-like)

```
t0==typeof(x);
var x=3;
var t1=typeof(x);
x="pippo";
var t2=typeof(x);
```

t0: undefined

t1: number

t2: string

Data types: Objects and DOM

Javascript also has Objects, somehow similar to Java Objects (even though their implementation is quite different, and their definition not as clean and straightforward as in Java).

Objects have variables and methods.

Similarly to Java Objects, they can be printed: in such case they use their customized toString() method, or give a generic indication such as **[object HTMLDivElement]**

Some Javascript Objects represent fragments of an HTML document. The collection of these Objects represent the whole page. Such representation is called Document Object Model.

Operators

- Mathematical operators: standard, plus ** for exponentiation (ES6)
- Assignment operators: standard, plus **= for exponentiation (ES6)
- String operators: + (concatenation), see later
- Comparison operators: standard, plus type and value

===	equal value and equal type
!==	not equal value or not equal type

- Logical and bitwise operators: standard
- Type operators

typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

see https://www.w3schools.com/js/js_operators.asp

More here:

https://www.w3schools.com/js/default.asp

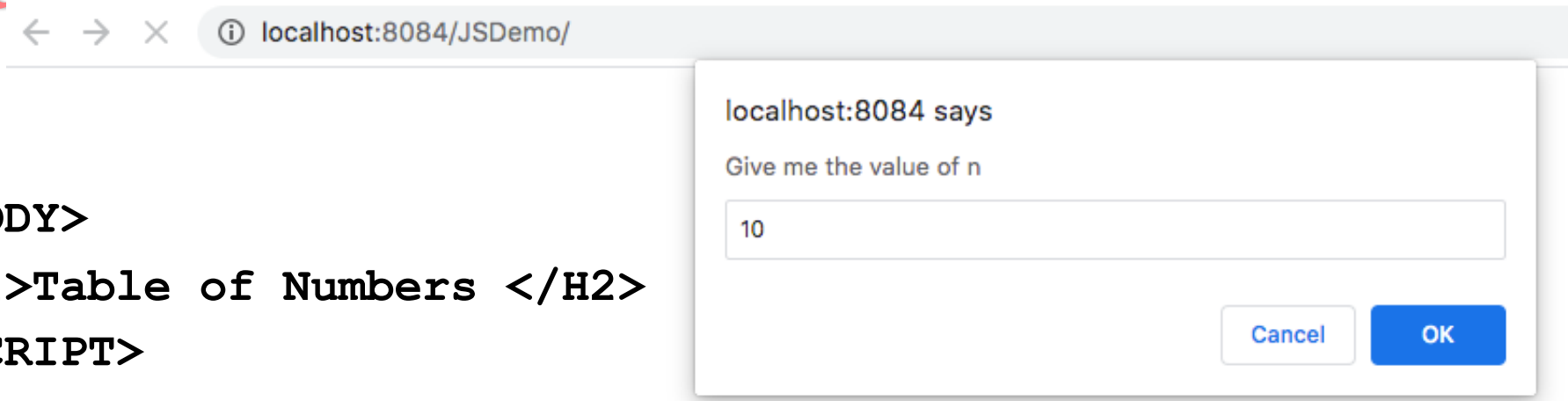
homework: go through these items!

Javascript and the DOM

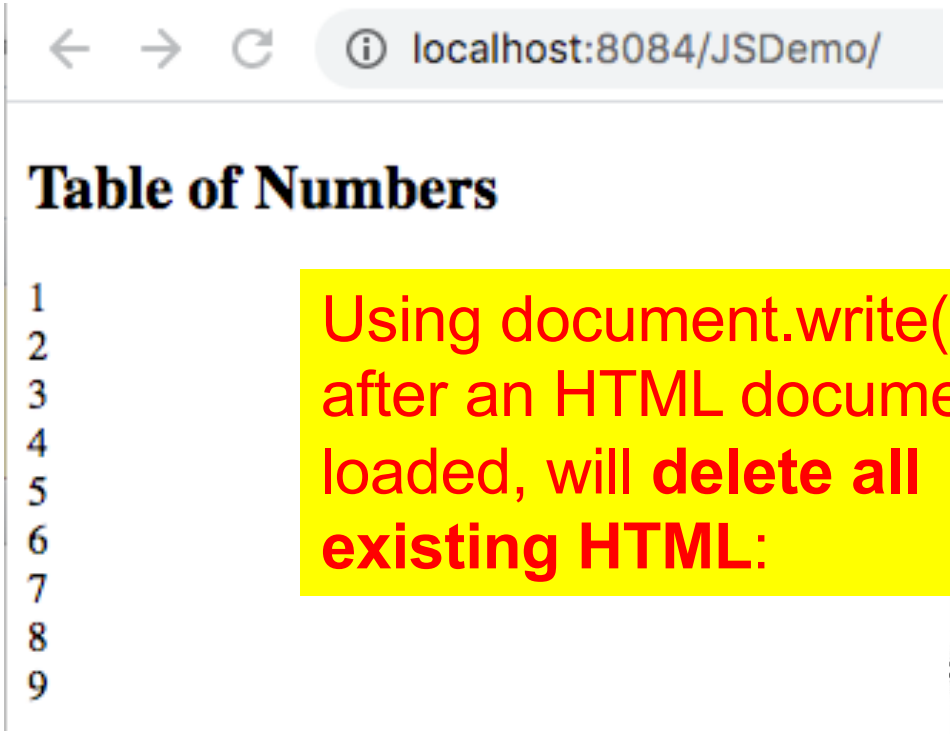
3- user input and output

Core

User I/O




```
<BODY>
<H2>Table of Numbers </H2>
<SCRIPT>
n=window.prompt("Give me the value of n",3);
for (i=1; i<n; i++) {
   document.write(i);
   document.write("<BR>");
}
</SCRIPT>
</BODY>
</HTML>
```



The dynamic behaviour is on the client side!
(The file can be loaded locally)

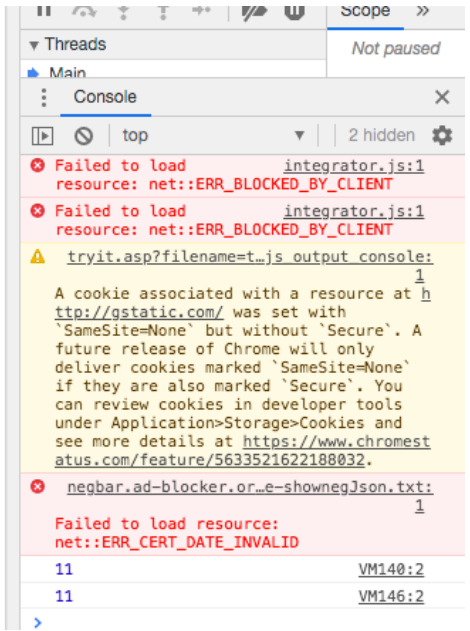
Using document.write() after an HTML document is loaded, will **delete all existing HTML**:

JS output

Writing into the HTML output using `document.write()`.

Writing into an alert box, using `window.alert()`.

Writing into the browser console, using `console.log()`.



JS output

Writing into an HTML element, using:

- innerHTML

```
<div onmouseover="this.innerHTML='How are you?';">
  Hello</div>
```

- innerText

```
<div onmouseover="this.innerText='How are you?';">
  Hello</div>
```

- textContent

```
<div onmouseover="this.textContent='How are you?';">
  Hello</div>
```

```
<div onmouseover="window.alert(this.property)";>
This element contains <code>code</code>, <span
   style="visibility:hidden">hidden information, </span>
   and <strong>strong language</strong>.</div>
```

This element contains code, and **strong language.**

- innerHTML

  Full HTML content

property

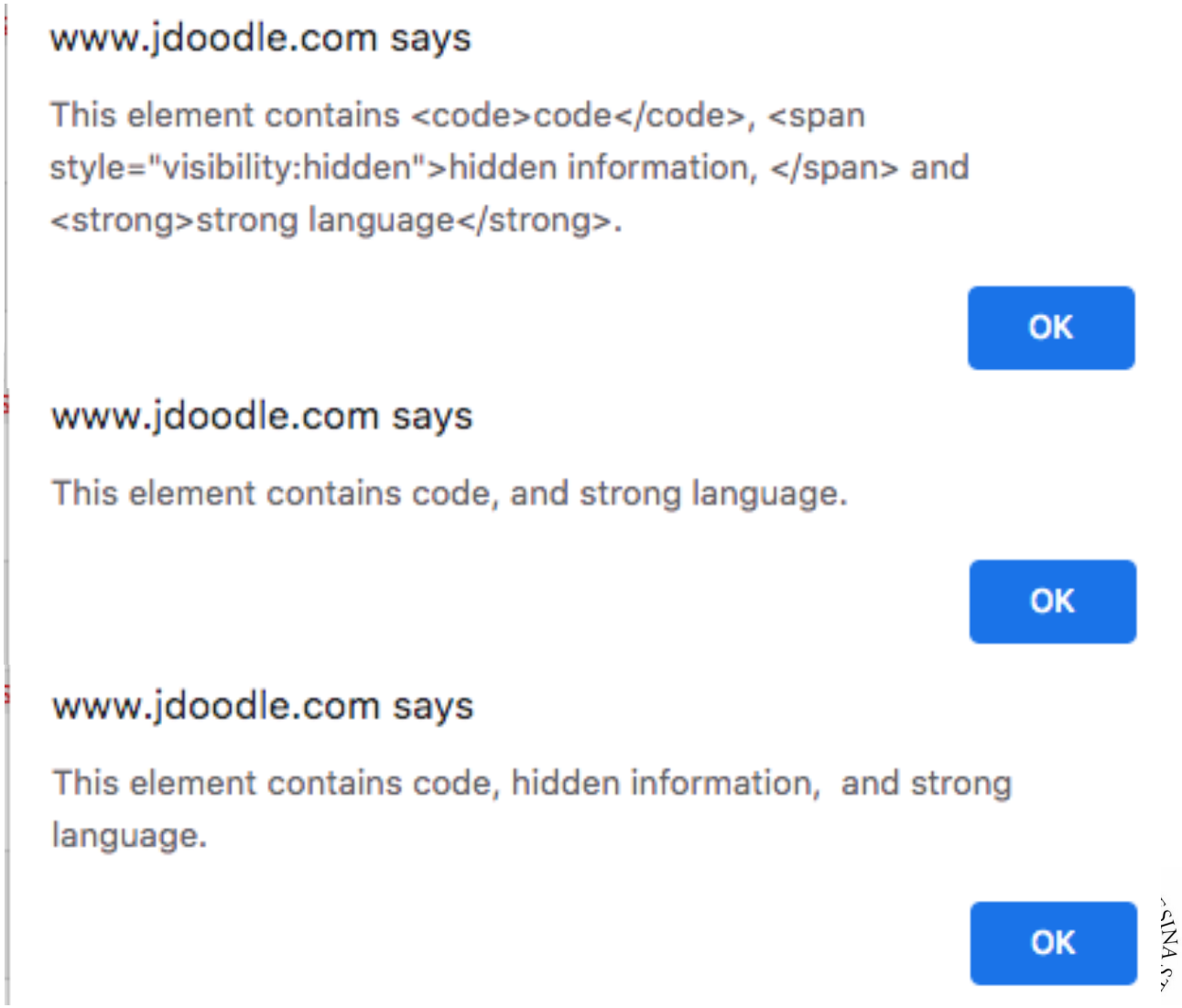
- innerText

  Text only, CSS aware

- textContent

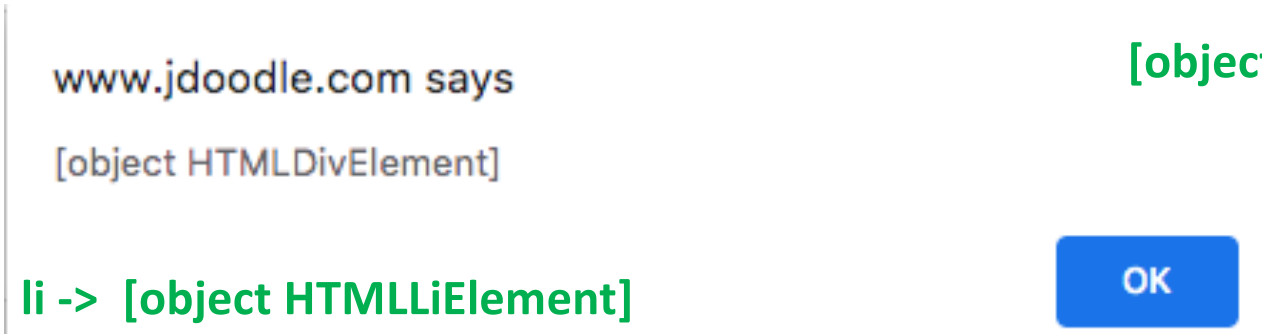
  Text only, CSS unaware

JS output



What is "this"?

```
<div onmouseover="window.alert(this)";"> Hello</div>
```



[object HTMLDivElement]

li -> [object HTMLLiElement]

h1,…h5 -> [object HTMLHeadingElement]

b, i -> [object HTMLElement]

```
<a onmouseover="window.alert(this)";"
  href="http://localhost"> a link</a>
```

a -> http://localhost

```
<b onmouseover="window.alert(this.nodeName)";"> Hello</b>
```

b -> b

a -> a