Design and Implementation of Online Behavioral Experiments

nodeGame.org

Stefano Balietti

MZES and Heidelberg

Advanced JavaScript

@balietti
@nodegameorg
stefanobalietti.com@gmail.com

Variable Declaration in JS

```
var a = 1; // number
var b = 'Hello world!'; // string
var c = function(p) { return p+1; }; // function
var d = { key: 'value' }; // object
var e = [ "value1", 3, c ]; // array
```

- Variables are **loosely typed**
- Variables are scoped within the **function** in which they are declared
- Functions and array are objects
- All the variables declared above are **global**

Variable Declaration in JS

```
var a = 1; // number
var b = 'Hello world!'; // string
var c = function(p) { return p+1; }; // function
var d = { key: 'value' }; // ob
var e = [ "value1", 3, c ]; //
```

Note! ES6 (the newest version of JavaScript) introduced some changes here. However, we stick to version ES5 to maximize our experimental audience.

- Variables are **loosely typed**
- Variables are scoped within the **function** in which they are declared
- Functions and array are objects
- All the variables declared above are **global**

Variable Declaration in JS

How to make variables *private* ?

Declare them inside an own function!

This technique is called: *closure*

```
function foo(bar) {
    var a = bar;
}
foo(10);
console.log(a); // undefined
```

Variable Declaration in JS

```
function foo() {
    var a = 1;
}
foo();
console.log(a); // undefined
```

What happens is we do not use the `var` keyword?

Variable Declaration in JS

```
function foo() {
    var a = 1;
}
foo();
console.log(a); // undefined
```

What happens is we do not use the `var` keyword?

JS will try to access the *global variable* `a`

Variable Declaration in JS

```
function foo() {
    var a = 1;
}
foo();
console.log(a); // undefined
```

What happens is we do not use the `var` keyword?

JS will try to access the *global variable* `a`
What if there is no *global variable* `a`?

Variable Declaration in JS

```
function foo() {
    var a = 1;
}
foo();
console.log(a); // undefined  1
```

What happens is we do not use the `var` keyword?

JS will try to access the *global variable* `a`
What if there is no *global variable* `a`?

Variable *leaking* into the global scope

Avoid Programming Mistakes

There are ways to avoid these types of mistakes in JS

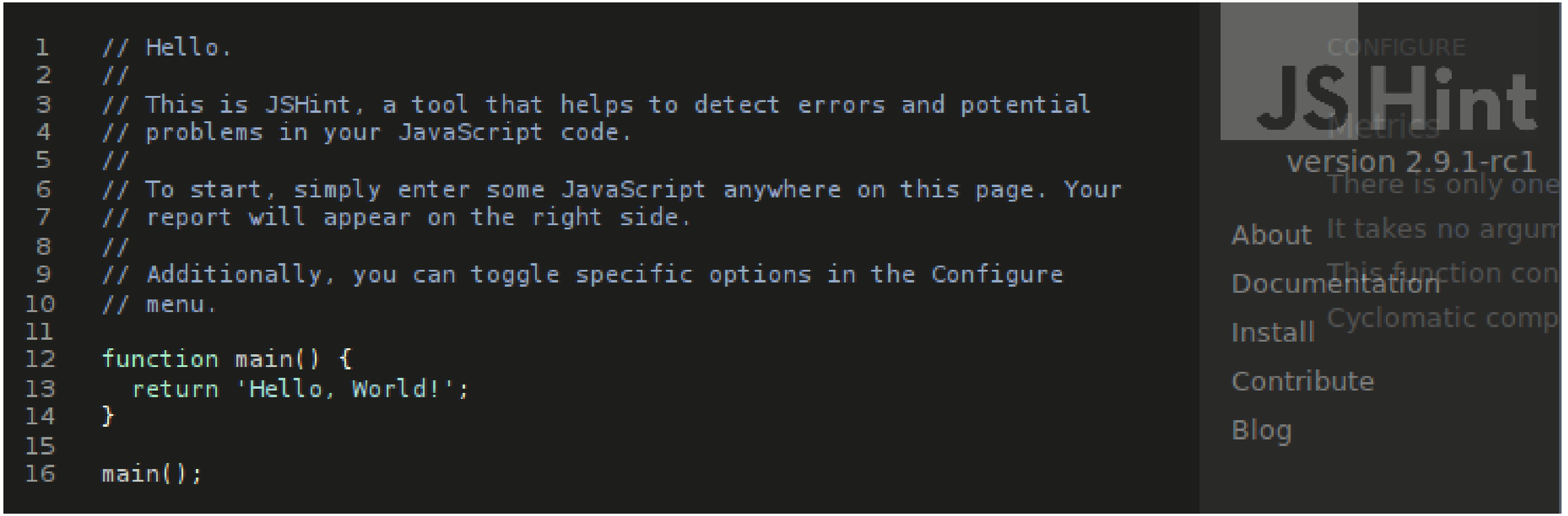
- Use a linting tool: JSHint/JSLint/ESLint/JSCS
  https://www.sitepoint.com/comparison-javascript-linting-tools

- JS Strict Mode

```
"use strict";
```

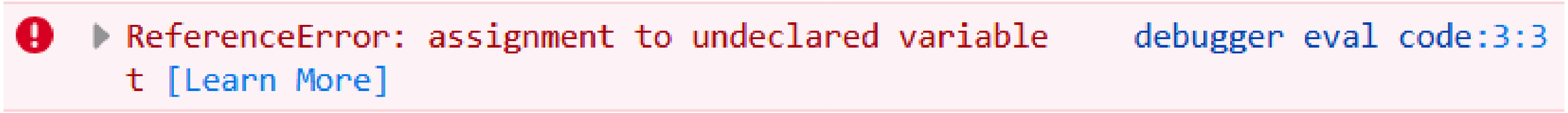
Linting Tool: JSHint

Try it on : http://jshint.com



Strict Mode on

```
function foo(bar) {
    "use strict";
     t = bar;
}
foo(10);
```



JS Function Input Parameters

```
// Internally modifies input.
function doSomething(obj, num, str) {
    obj.a = 10;
    num = 1;
    str = 'a';
}
var obj = {}, num = 0, str = '';
doSomething(obj, num, str);

console.log(obj);
console.log(num);
console.log(str);
```

What will the final values of the object, the string, and the number be, after they have been modified by the function?

JS Function Input Parameters

```
// Internally modifies input.
function doSomething(obj, num, str) {
    obj.a = 10;
    num = 1;
    str = 'a';
}
var obj = {}, num = 0, str = '';
doSomething(obj, num, str);

console.log(obj); // { a: 10 }
console.log(num); // 0
console.log(str); // ''
```

Objects are passed as a *reference (to an address in memory),* while numbers and strings are *copies*

Modifying a copy does not affect the value outside the function, modifying the reference does.

Our Previous Example: Arrays and For Loops

```
var message  = 'I like ';
// This is a "for loop".
for (var i = 0 ; i < fruits.length ; i++) {
    message += fruits[i];
    if (i < (fruits.length - 1 )) {
        message += ', ';
    }
    else {
        message += '.';
    }
}
alert(message);
```

Our Previous Example: Arrays and For Loops

```
var message  = 'I like ';
// This is a "for loop".
for (var i = 0 ; i < fruits.length ; i++) {
    message += fruits[i];
    if (i < (fruits.length - 1 )) {
        message += ', ';
    }
    else {
        message += '.';
    }
}
alert(message);
```

That's a lot of code inside the for-loop. How to make it more compact?

Functions with Returns

```
function join(word, index, arraySize, endSign) {
    if (undefined === endSign) endSign = '.';
    if (index === arraySize -1) word += ',';
    else word += endSign;
    return word;
}
```

If-else branches can be written without parentheses, and they apply to the next line, as delimited by semicolon (;).

```
var message  = 'I like ';
// This is a "for loop".
for (var i = 0 ; i < fruits.length ; i++) {
   message += join(fruits[i], i, fruits.length, "!");
}
```

Functions with Returns

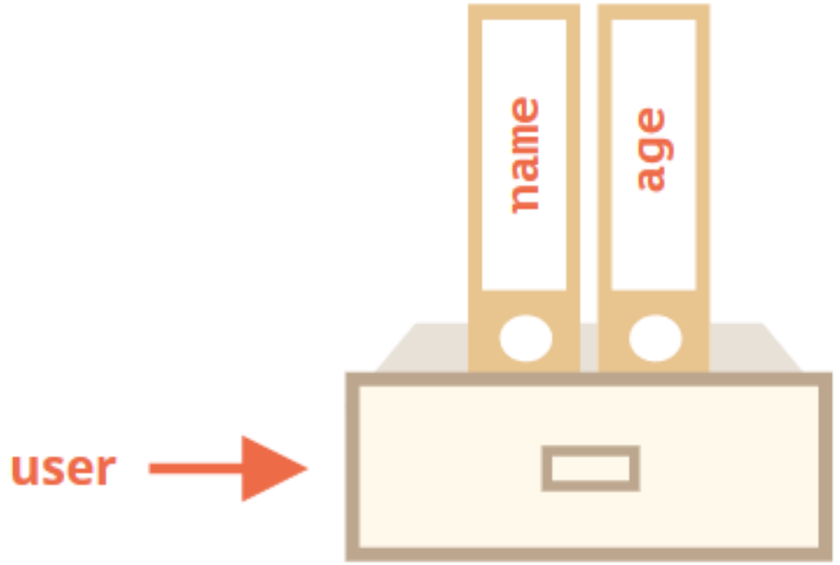
```
function join(word, index, arraySize, endSign) {
    if (undefined === endSign) endSign = '.';
    if (index === arraySize -1) word += ',';
    else word += endSign;
    return word;
}

var message  = 'I like ';
// This is a "for loop".
for (var i = 0 ; i < fruits.length ; i++) {
   message += join(fruits[i], i, fruits.length, "!");
}
```

The return keyword makes available outside of the function the modified variable `word`.

Functions are Objects

http://javascript.info/

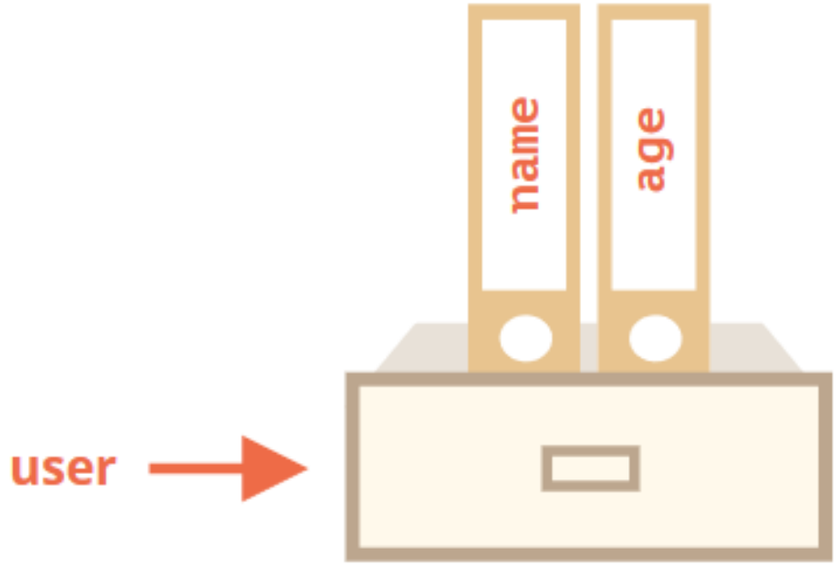
```
// Our previous example of user John.
var user = {
  name: "John", // by key "name" store value "John"
  age: 30       // by key "age" store value 30
};
```



Functions are Objects

http://javascript.info/

```
// Our previous example of user John.
var user = {
  name: "John", // by key "name" store value "John"
  age: 30       // by key "age" store value 30
};
// It is equivalent to:
function User() {
  this.name = "John";
  this.age = 30;
}
var user = new User();
```



Creating Objects with a Constructor

http://javascript.info/

```
// Our previous example of user John.
var user = {
  name: "John", // by key "name" store value "John"
  age: 30       // by key "age" store value 30
};
// It is equivalent to:
function User() {
  this.name = "John";
  this.age = 30;
}
var user = new User();
```

In the technical language the function User is called the "constructor" or "class," and the variable user is the live "instance" of the class User.

Creating Objects with a Constructor

```
// Our previous example of user John.
var user = {
  name: "John", // by key "name" store value "John"
  age: 30       // by key "age" store value 30
};
// It is equivalent to:
function User() {
  this.name = "John";
  this.age = 30;
}
var user = new User();
```

Behind the scenes, "new" is adding the lines in bold to the User function

```
function User() {
  var this = {};
  this.name = "John";
  this.age = 30;
  return this;
}
```

Creating Objects with a Constructor

```
// Our previous example of user John.
var user = {
  name: "John", // by key "name" store value "John"
  age: 30       // by key "age" store value 30
};
// It is equivalent to:
function User() {
  this.name = "John";
  this.age = 30;
}
var user = new User();
```

What is the advantage of using a constructor function?

Constructors Take Input Parameters

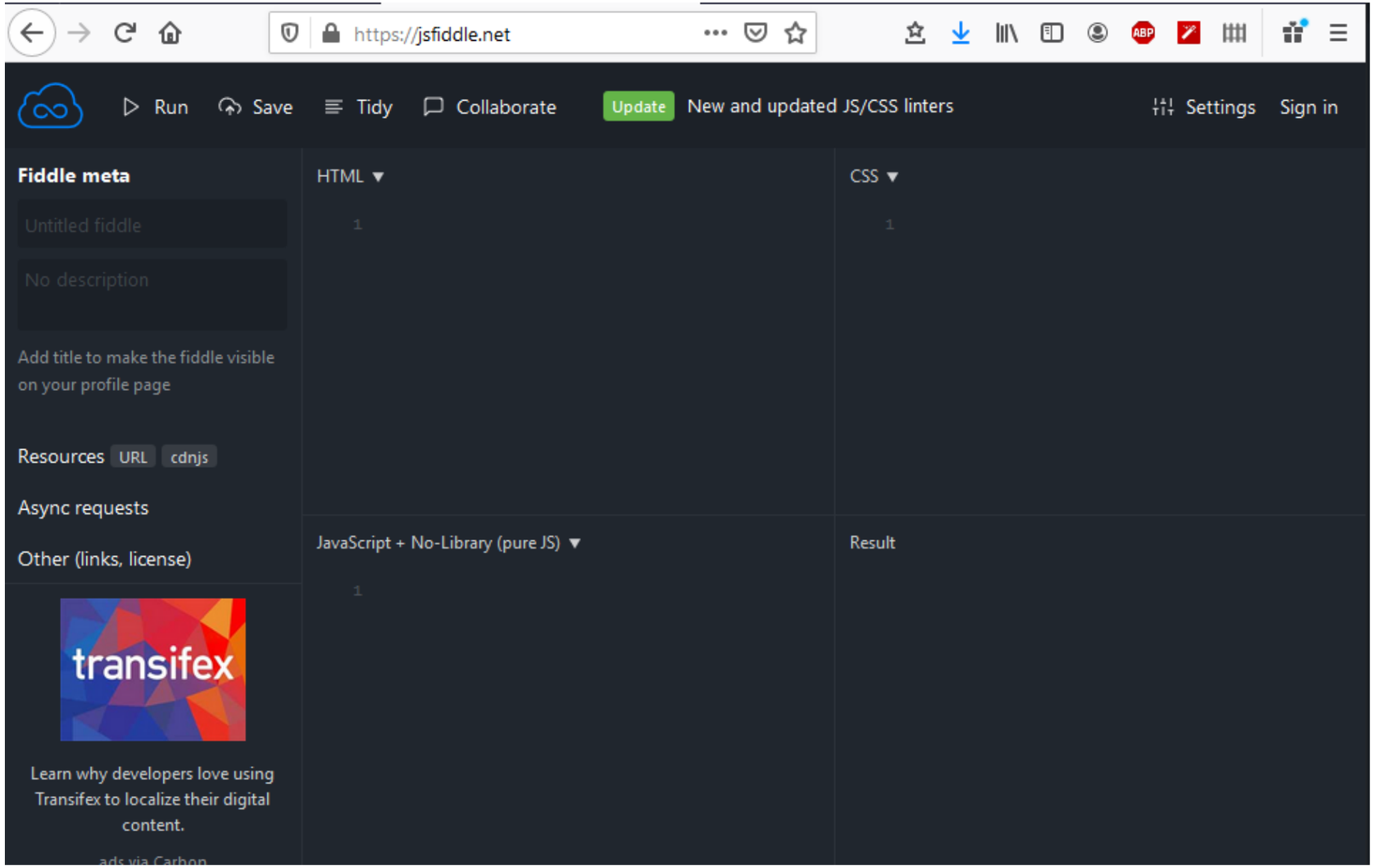
```
function User(name, age) {
  this.name = name;
  this.age = age;
}
var user = new User("John", 30);
```

Constructors Can Define Methods

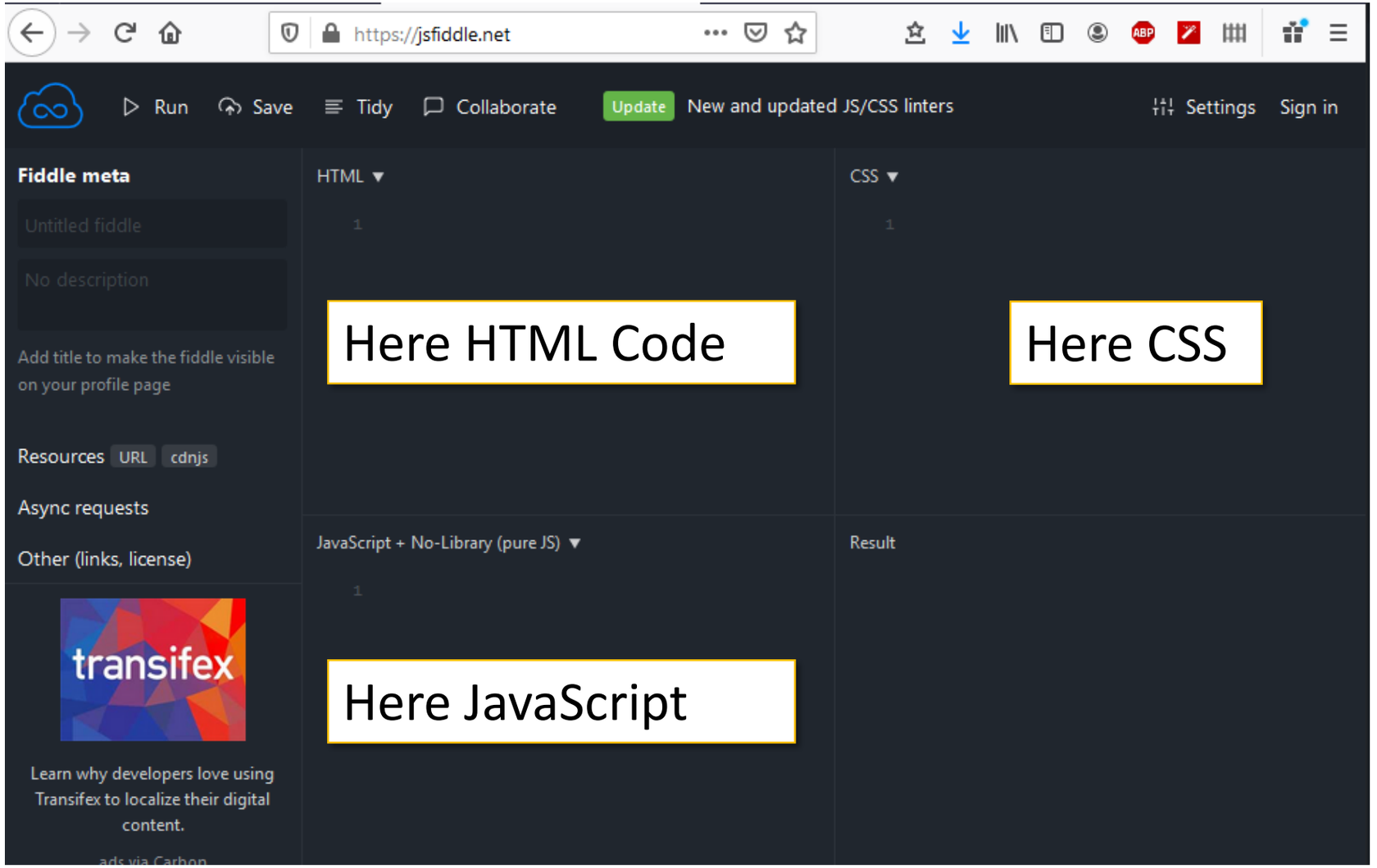
```
function User(name, age) {
   this.name = name;
   this.age = age;

   this.sayHello = function(to) {
        var hello = 'Hello ' + to;
        hello += '. I am ' + this.name;
        hello += ', and I am ' + this.age;
        alert(hello);
   };
}
```

The order in which properties and methods inside of the constructor are defined does not matter. They are evaluated only after the constructor is *instantiated* into a live object.

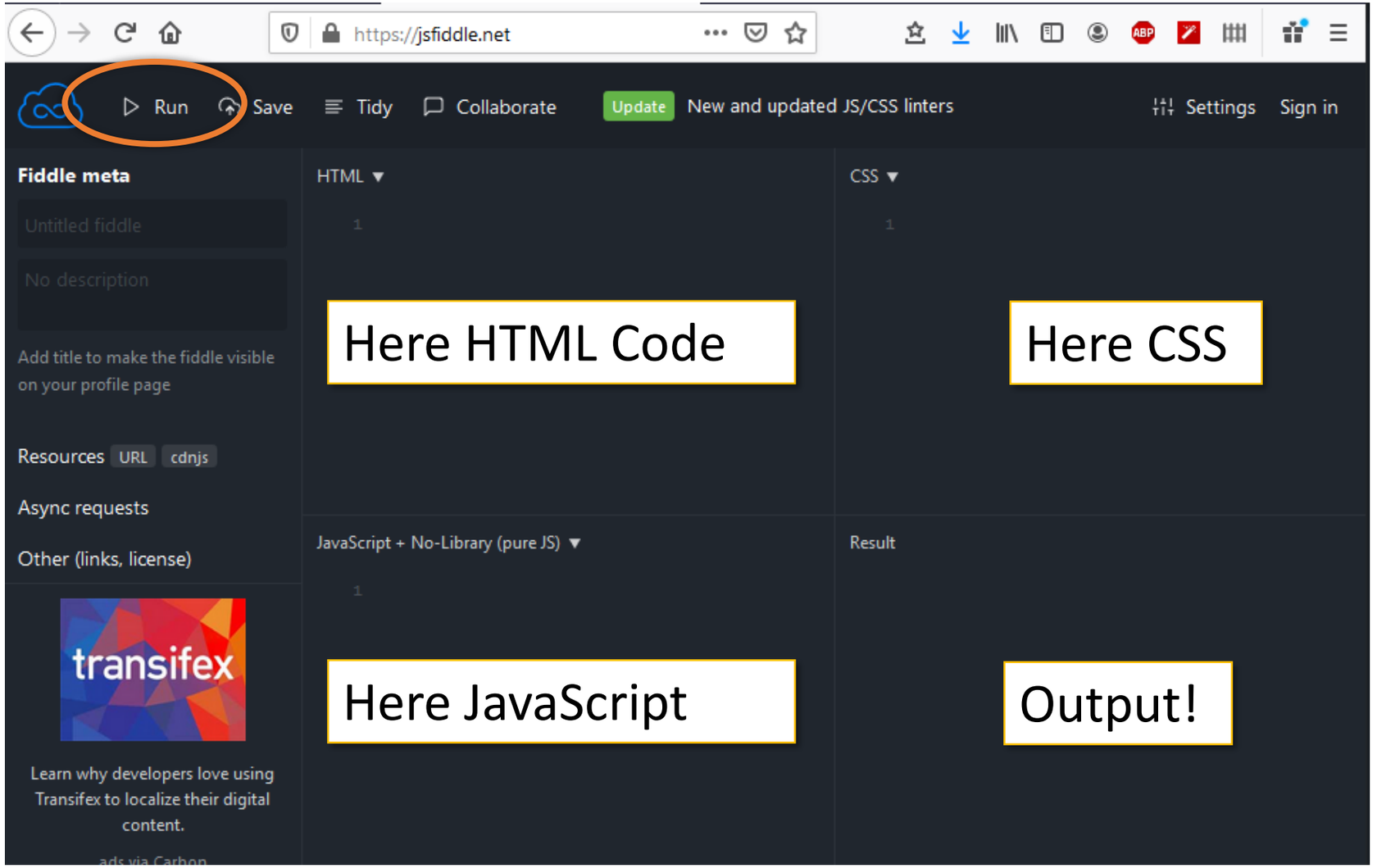
JS Fiddle


https://jsfiddle.net
Run
Save
Tidy
Collaborate
Update
New and updated JS/CSS linters
Settings
Sign in
Fiddle meta
Untitled fiddle
No description
Add title to make the fiddle visible on your profile page
Resources
URL
cdnjs
Async requests
Other (links, license)
HTML
CSS
JavaScript + No-Library (pure JS)
Result
transifex
Learn why developers love using Transifex to localize their digital content.
ads via Carbon

JS Fiddle


https://jsfiddle.net
Run
Save
Tidy
Collaborate
Update
New and updated JS/CSS linters
Settings
Sign in
Fiddle meta
Untitled fiddle
No description
Add title to make the fiddle visible on your profile page
Resources
URL
cdnjs
Async requests
Other (links, license)
HTML
CSS
Here HTML Code
Here CSS
JavaScript + No-Library (pure JS)
Result
Here JavaScript
transifex
Learn why developers love using Transifex to localize their digital content.
ads via Carbon

JS Fiddle


https://jsfiddle.net
Run
Save
Tidy
Collaborate
Update
New and updated JS/CSS linters
Settings
Sign in
Fiddle meta
Untitled fiddle
No description
Add title to make the fiddle visible on your profile page
Resources
URL
cdnjs
Async requests
Other (links, license)
HTML
CSS
JavaScript + No-Library (pure JS)
Result
Here HTML Code
Here CSS
Here JavaScript
Output!
transifex
Learn why developers love using Transifex to localize their digital content.
ads via Carbon

Hands On 4 A

1. Add two inputs (age and name)

hint: HTML tag <input type="text" />; check the "placeholder" attribute)

2. Add one button

(hint: HTML tag <button>Text</button>)

3. When the button is clicked a new object of type User is created with values from the two inputs.

(hints: get the inputs with document.getElementById and look into the property named value; remember the onclick function?)

4. Display an alert message of the successful creation

Hands On 4 B

5. Add another input for *your* name and another button
6. When the button is clicked,
   A. The method sayHello of the object of class User is invoked with the value of the input as parameter.
   B. The return value of sayHello is added to the page.

   Hints: document.createElement; document.body.appendChild; innerHTML property

Solution

Yes, I clicked the "Press Me" button a few times.

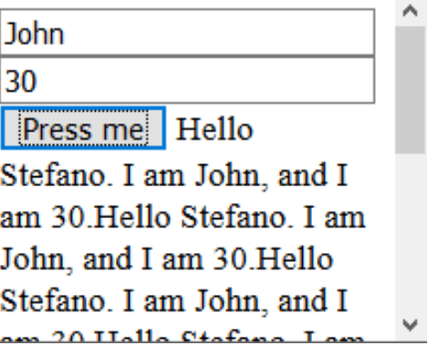
HTML

```html
<input type="text" id="myname" placeholder="Name"/>
<input type="text" id="myage" placeholder="Age"/>
<button id="mybutton">Press me</button>
```

CSS

JavaScript + No-Library (pure JS)

```javascript
function User(name, age) {
  this.name = name;
  this.age = age;
  this.sayHello = function(to) {
      var hello = 'Hello ' + to;
      hello += '. I am ' + this.name;
      hello += ', and I am ' + this.age + '.';
      // alert(hello);
      return hello;
  };
}

var name = document.getElementById("myname");
var age = document.getElementById("myage");
var button = document.getElementById('mybutton');

button.onclick = function() {
   // Example of input validation.
   if (isNaN(parseInt(age.value, 10))) {
      alert('Hey this is Not A Number (NaN)!')
      // Return to avoid execution of the code below.
      return;
   }
   // Assume inputs are valid.
   var user = new User(name.value, age.value);
   var hello = user.sayHello("Stefano")
   var span = document.createElement(span);
   span.innerHTML = hello;
   document.body.appendChild(span);
}
```



Bonus Hands On 4 C

7. Assign a random color to every new DIV containing the result of sayHello

Hints:

create an array of colors = ["red", "green", …];

get a random index from 0 to the size of the array - 1;

Math.floor(Math.random() * myArray.length);