JavaScript

Very Basic

2015-02-21

Data Type

How many data types in JavaScript?

- Number (Integer and Float)
- String (Text or Number as text)
- Boolean (True or False)
- Undefined (Didn’t assign a value on it)
- Infinity and –Infinity
- NaN

Number Type 1

Number (64 bit or 8 bytes , range 2^{64})

- Can be integer and fraction
- Not very precise for fraction (EX: Currency App) but reliable for integer.

```
var numData = 1;
var numData = 1.1;
```

Number Type 2

For newbie avoid this

```
var octal = 0377;
var hex = 0xFF;
```

What can you do with number?

```
x = x+5; // is the same as x += 5;
x = x-5; // is the same as x -= 5;
x = x*5; // is the same as x *= 5;
x = x/5; // is the same as x /= 5;
x = x%5; // is the same as x %= 5;
```

Number Type 3

```
x += y;    // is the same as x = x + y
x -= y;    // is the same as x = x - y
x *= y;    // is the same as x = x * y
x /= y;    // is the same as x = x / y
x %= y;    // is the same as x = x % y
x++;       // is the same as x = x + 1
x--;       // is the same as x = x - 1
```

String Type 1

Store text or number of text.

```
var myText1 = “Sample Text”;
var myText2 = ‘Sample Text’;
var myText3 = “Sample\\n\“Data”\”;
var myText4 = “45”;
var myText5 = “3 + 3”;
```

Append String

```
var tmp = “word: ”;
tmp = “app” + “end” + “ ” + “string”;
```

String Type 2

What about substring?

```
var str = "Hello world!";
var res = str.substring(1, 4);
// ell
```

What about replace?

```
var str = "Visit Microsoft!";
var res = str.replace("Microsoft", “Apple");
// Visit Apple!
```

What about empty string?

```
var str = “This is not empty string”;
str = “”;
```

Boolean Type 1

Contain just only true or false

```
var tmp = true;
tmp = (10 < 9);
```

How can you get Boolean value

Comparison and condition

`==`	equal to	`if (day == "Monday")`
`>`	greater than	`if (salary > 9000)`
`<`	less than	`if (age < 18)`

Boolean Type 2

Logical Operator

```
console.log(true && false)
// → false
console.log(true && true)
// → true

console.log(false || true)
// → true
console.log(false || false)
// → false


console.log("Itchy" != "Scratchy")
// → true
```

Logical operators also include ===, !==, ==, !=, >=, <=, >, <, and !.

Others Data Type 1

Infinity and –Infinity

- Infinity is displayed when a *number exceeds the upper limit of the floating point numbers*, which is 1.797693134862315E+308.

- -Infinity is displayed when *a number exceeds the lower limit of the floating point numbers*, which is -1.797693134862316E+308.

- Infinity – 1 equivalent to infinity.

- If you calculate value with infinity, it can result in Not a Number (NaN).

Others Data Type 2

Not a Number (NaN)

- Happen when you perform “0 / 0” or Infinity – Infinity and etc.

```
console.log(0 / 0);
// → NaN

console.log(NaN == NaN);
// → false

isNaN(value);
```

Undefined

- You *create a variable but didn’t assign a value to it*. So, it does not know what type of the value is and cannot assign a default value to it.

Automatic Type Conversion 1

```
console.log(8 * null)
// → 0

console.log("5" - 1)
// → 4

console.log("5" + 1)
// → 51

console.log("five" * 2)
// → NaN

console.log(false == 0)
// → true
```

To avoid automatic type conversion not to happen, there are two extra operators === and !==.

Automatic Type Conversion 2

==	equal to	x == 8	false
		x == 5	true
===	equal value and equal type	x === "5"	false
		x === 5	true
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== "5"	true
		x !== 5	false

Null Value

When null or undefined occurs on either side of the operator, it produces true only if both sides are one of null or undefined.

```
console.log(null == undefined);

// → true

console.log(null == 0);

// → false
```

Therefore, when you want to test weather a value has areal value instead of null or undefined, you can simply compare it to null with the == or != operator.

Function

```
console.log(square (12));
// → 144

var theNumber = Number(prompt("Pick a number", ""));
alert("Your number is " + theNumber * theNumber);
```

Defining a Function

```
var square = function(x) {
       return x * x;
};


Or


function square(x){

       Return x * x;

};

console.log(square (12));
// → 144
```

Parameters and Scope

```
var x = "outside";
var f1 = function()
{
       var x = "inside f1";
};
f1();
console.log(x);
// → outside


var f2 = function() {
       x = "inside f2";
};
f2();
console.log(x);
// → inside f2
```

Nested Scope

```
var landscape = function() {
       var result = "";
       var flat = function(size) {
              for (var count = 0; count < size; count++)
                     result += "_";
       };
       var mountain = function(size) {
              result += "/";
              for (var count = 0; count < size; count++)
                     result += "'";
              result += "\\";
       };
       flat(3);
       mountain(4);
       flat(6);
       mountain(1);
       flat(1);
       return result;
};
console.log(landscape());

  // → ___/''''\______/'\
```

Namespace in JavaScript

JavaScript, functions are the only things that create a new scope.

Optional Arguments

```
function power(base , exponent) {
        if (exponent == undefined)
                exponent = 2;

        var result = 1;
        for (var count = 0; count < exponent; count++)
                result *= base;

        return result;
}

console.log(power(4));
// → 16

console.log(power(4, 3));
// → 64
```

Higher-Order Function 1

```
function forEach(array , action) {
        for (var i = 0; i < array.length; i++)
                action(array[i]);
}

forEach(["Wampeter", "Foma", "Granfalloon"], console.log);

// → Wampeter
// → Foma
// → Granfalloon
```

Higher-Order Function 2

```
function greaterThan(n) {
	return function(m) { return m > n; };
}

var greaterThan10 = greaterThan (10);

console.log(greaterThan10 (11));

// → true
```

Array (List)

Array/List/Stack/Queue

```
var listOfNumbers = [2, 3, 5, 7, 11];

console.log(listOfNumbers [1]);
// → 3
console.log(listOfNumbers[1 - 1]);
// → 2
```

Stack

```
var mack = [];
mack.push("Mack");
mack.push("the", "Knife");

console.log(mack);
// → ["Mack", "the", "Knife"]

console.log(mack.join(" "));
// → Mack the Knife

console.log(mack.pop());
// → Knife

console.log(mack);
// → ["Mack", "the"]
```

Queue

```
var arr = [1, 2];

arr.unshift(0); // result of call is 3, the new array length
// arr is [0, 1, 2]

arr.unshift(-2, -1); // = 5
// arr is [-2, -1, 0, 1, 2]

arr.unshift([-3]);
// arr is [[-3], -2, -1, 0, 1, 2]

arr.shift();
// [-3]
// arr is [-2, -1, 0, 1, 2]
```

Object

```
var day1 = {
       squirrel: false ,
       events: ["work", "touched tree", "pizza",
       "running","television"]
};

console.log(day1.squirrel);
// → false

console.log(day1.wolf);
// → undefined

day1.wolf = false;
console.log(day1.wolf);
// → false
```

Object Methods

```
var whiteRabbit = {type: "white", speak: speak};

function speak(line) {
       console.log("The " + this.type + " rabbit says '" +
       line + "'");
}



whiteRabbit.speak("Oh my ears and whiskers , " +
"how late it's getting!");

// → The white rabbit says 'Oh my ears and whiskers , how
// late it's getting!'
```

Object Constructor

```
function Rabbit(type) {
         this.type = type;
         this.speak = function(word){
                  console.log("Say " + word + " by " + this.type);
         }
}
var killerRabbit = new Rabbit("killer");
var blackRabbit = new Rabbit("black");
console.log(blackRabbit.type);

// black
killerRabbit.speak("hahaha");

// Say hahaha by killer
```

JSON

```
[
        { "name": "Emma de Milliano", "sex": "f"},
        {"name": "Carolus Haverbeke", "sex": "m"}
]
```

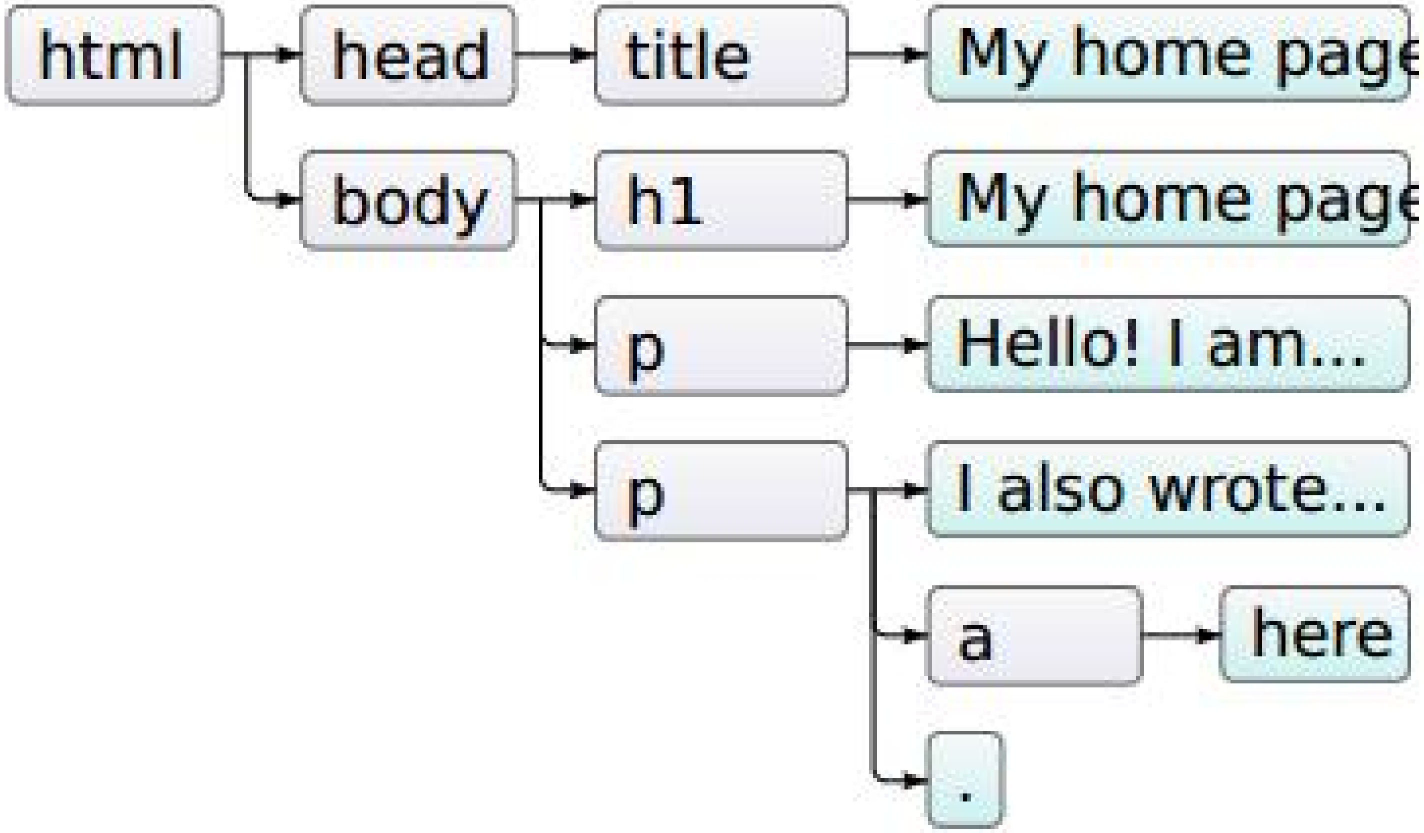
The Document Object Model

(DOM)

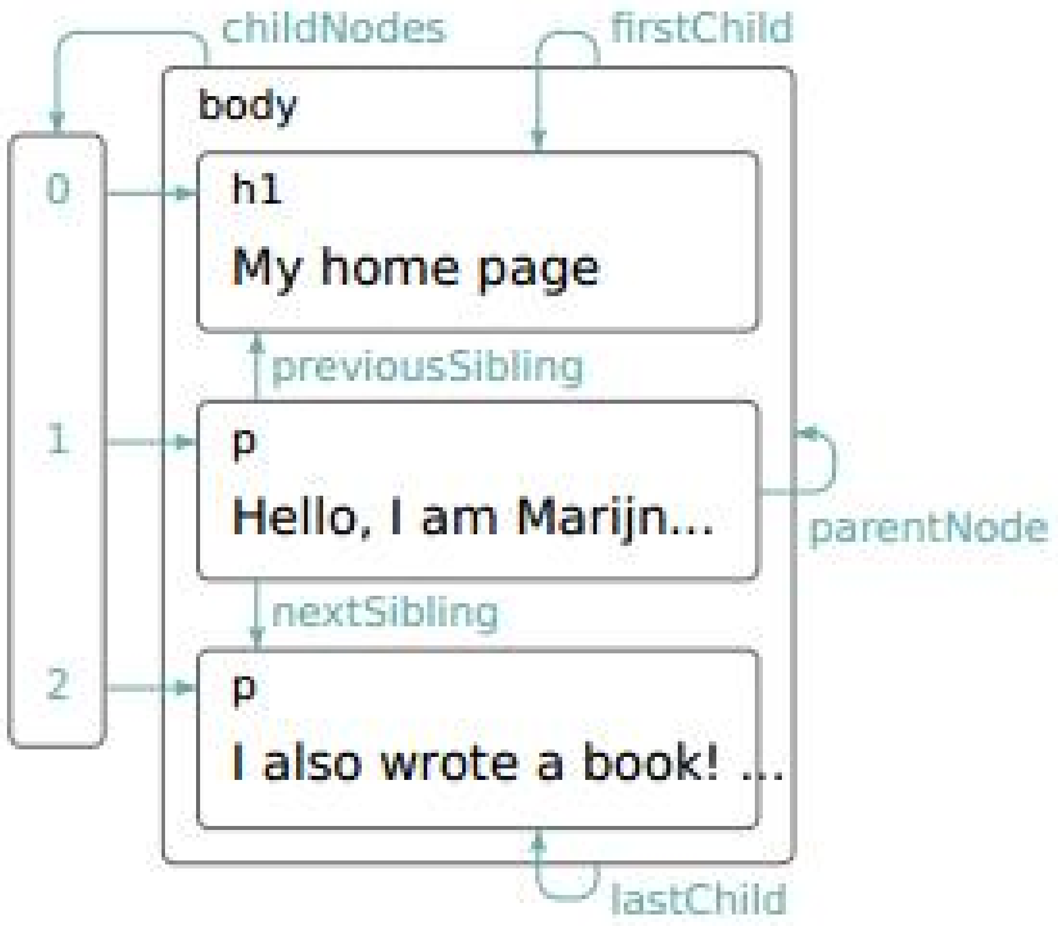
HTML

```
<!doctype html >
<html >
    <head >
        <title >My home page </title >
    </head >
    <body >
        <h1>My home page </h1>
        <p>Hello , I am Marijn and this is my home page.</p>
        <p>I also wrote a book! Read it
        <a href="http:// eloquentjavascript.net">here
</a>.</p>
    </body >
</html >
```

DOM Tree


html
head
title
My home page
body
h1
My home page
p
Hello! I am...
p
I also wrote...
a
here
.

Moving through the tree



Finding Elements

```
document.getElementById("ObjectId")
document.getElementsByName("ObjectName");
document.getElementsByTagName("div");
document.getElementsByTagNameNS("nameSpace","searchTag");


var link = document.body.getElementsByTagName("a")[0];
console.log(link.href);
```

Get Element

```
<p>My ostrich Gertrude:</p>
<p><img id="gertrude" src="img/ostrich.png"></p>

<script>
   var ostrich = document.getElementById("gertrude");
   console.log(ostrich.src);
</script >
```

Change Element Content

```
HTML:

<h1 id="header">Old Header</h1>

JavaScript:

var element = document.getElementById("header");
element.innerHTML = "New Header";

-------------------------------------------------------------

HTML:

<img id="myImage" src="smiley.gif">

JavaScript:

document.getElementById("myImage").src = "landscape.jpg";
```

Change Element Style

```
HTML:
<p id="p2">Hello World!</p>

JavaScript:
document.getElementById("p2").style.color = "blue";
-------------------------------------------------------------
HTML:
<h1 id="id1">My Heading 1</h1>
JavaScript:
<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>
```

Binding Event 1

```
HTML:

<h1 onclick="changeText(this)">Click on this text!</h1>


JavaScript:
function changeText(id) {
    id.innerHTML = "Ooops!";
}
------------------------------------------------------------
HTML:

<input type="button" onclick="setTextBlack()" value="Black"
/>


JavaScript:

function setTextBlack()

{      output.style.color = "black";     }
```

Binding Event 2

```
HTML:

<button >Click me </button >
<p>No handler here.</p>


JavaScript:

var button = document.querySelector("button");
button.addEventListener("click", function() {
       console.log("Button clicked.");
});
```

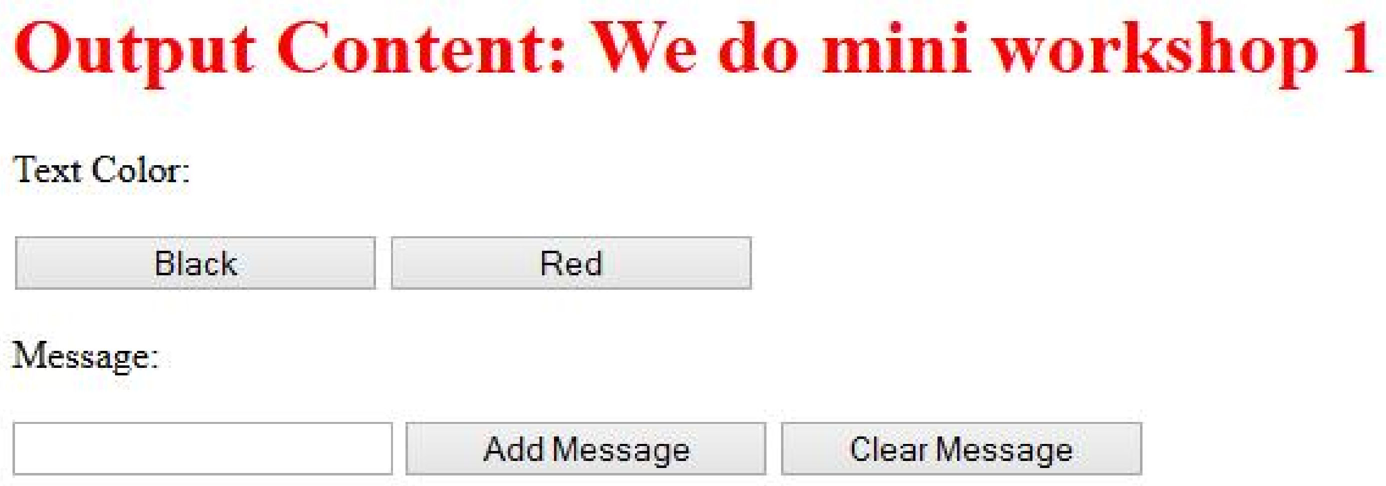
Note: querySelector() method returns the first element that matches a specified *CSS selector(s)* in the document.

MoreEvent: http://www.w3schools.com/jsref/dom_obj_event.asp

Unbinding Event

```
HTML:

<button >Act -once button </button >


JavaScript:

var button = document.querySelector("button");
function once() {
    console.log("Done.");
    button.removeEventListener("click", once);
}

// You can bind it back
button.addEventListener("click", once);
```

Mini Workshop 1



Output Content: We do mini workshop 1

Text Color:

Message:

HTTP Request

```
function httpGet(theUrl)
{
    var xmlHttp = null;

    xmlHttp = new XMLHttpRequest();
    xmlHttp.open( "GET", theUrl, false );
    xmlHtt
p.send( null );
    return xmlHttp.responseText;
}
```

GET requests, we can pass null.

POST and GET

GET:

http://pub.jamaicann.net/fpdb/api.php?username=jorass&password=joras
s&action=inventory_get

POST:

http://pub.jamaicainn.net/fpdb/api.php

	GET(HTTP)	**Post(HTTP)**
History	Parameters remain in browser history because they are part of the URL	Parameters are not saved in browser history.
Bookmarked	Can be bookmarked.	Can not be bookmarked.

JSON Data

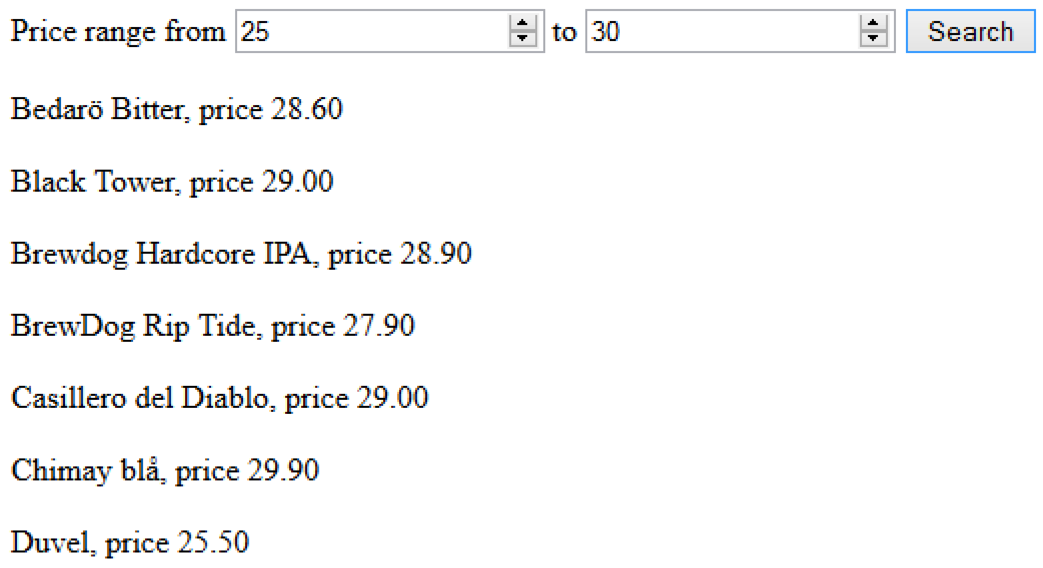
```
var result = httpGet(request);
var parseResult = JSON.parse(result);
----------------------
```

```
var data = parseResult['payload'];
```

```
{
    "type": "inventory_get",
    "payload": [
        {
            "namn": "Beck's",
            "namn2": "",
            "sbl_price": "14.90",
            "pub_price": "20",
            "beer_id": "154903",
            "count": "106",
            "price": "14.10"
        },………… ]
}
```

```
data[i].namn
```

Mini Workshop 2



Bedarö Bitter, price 28.60

Black Tower, price 29.00

Brewdog Hardcore IPA, price 28.90

BrewDog Rip Tide, price 27.90

Casillero del Diablo, price 29.00

Chimay blå, price 29.90

Duvel, price 25.50

Creating Element

```
var node = document.createElement("div");
node.setAttribute("name",data[i].beer_id);
node.setAttribute("class","listItem");

var drinkPrice = document.createElement("p");
drinkPrice.innerHTML = data[i].sbl_price + "kr";

node.appendChild(drinkImg);
node.appendChild(section);
```

Document States

One of five values:

- uninitialized - Has not started loading yet
- loading - Is loading
- loaded - Has been loaded
- interactive - Has loaded enough and the user can interact with it
- complete - Fully loaded

```
// alternative to DOMContentLoaded
document.onreadystatechange = function () {
  if (document.readyState == "interactive") {
    initApplication();
  }
}
```

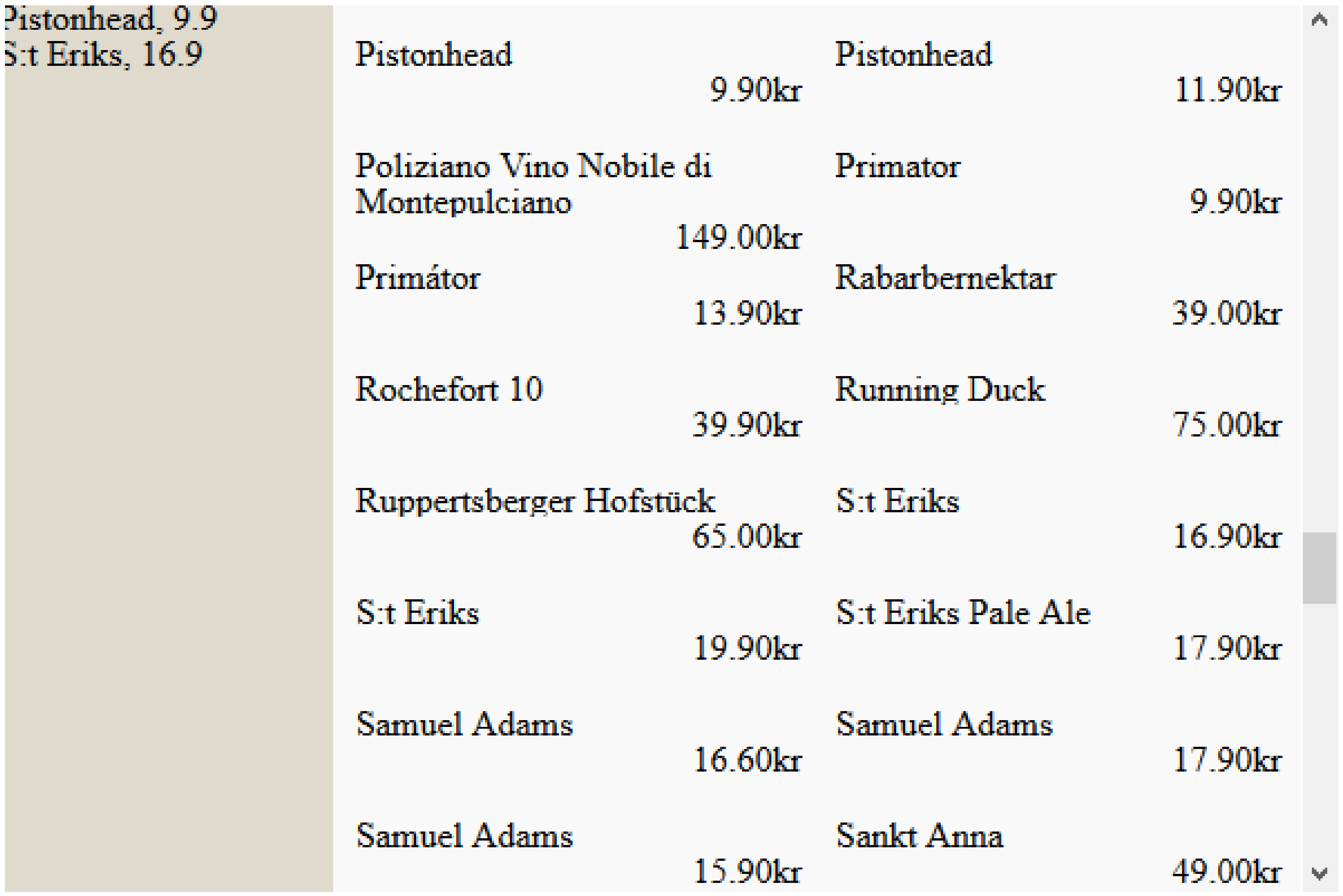
Window Event

Onload event occurs when all content has been loaded.

JavaScript:

```
window.onload = function()
{
       Init();
       doSomethingElse();
};

Or

HTML:
<body onload="myFunction()">
```

Mini Workshop 3



Pistonhead, 9.9
S:t Eriks, 16.9

Pistonhead 9.90kr	Pistonhead 11.90kr
Poliziano Vino Nobile di Montepulciano 149.00kr	Primator 9.90kr
Primátor 13.90kr	Rabarbernektar 39.00kr
Rochefort 10 39.90kr	Running Duck 75.00kr
Ruppertsberger Hofstück 65.00kr	S:t Eriks 16.90kr
S:t Eriks 19.90kr	S:t Eriks Pale Ale 17.90kr
Samuel Adams 16.60kr	Samuel Adams 17.90kr
Samuel Adams 15.90kr	Sankt Anna 49.00kr

Congratulation!