

# JavaScript Basics

---

Prof. Emanuele Panizzi

- just-in-time compiled language
- executed in the browser (but also some servers have JS engine)
- can access and modify the Document Object Model

- strings, numbers, booleans, ...
- objects (include arrays)
- dynamic typing (type of a variable can change)
- weakly typed (casting depending on operation)

```
var a;           // function-scoped
var b = 1;
b += 5;         // reassigned, now it's 6
let c;          // block-scoped
let d = 3/2;    // it's 1.5
d = "Hello";   // reassigned, changed type
d += " world"; // now "Hello world"
d = d + 1;     // now "Hello world1"
const k = 30;  // block-scoped, cannot be reassigned
x = "foo";     // deprecated assignment (global scope)
```

- collection of properties
- mutable
- normally:
  - property keys are strings
  - property values are of any js type
- Array, Function, Date, RegExp, Error: they are objects

## Arrays

- not a primitive but an object
- indexes are strings converted from integer numbers, from 0
- brackets notation `[]`

```
const rooms = ["1L", "2L", "3L"] // 1L,2L,3L
rooms[0] = "1LL"; // 1LL,2L,3L
rooms[4] = "MeetingRoom"; // 1LL,2L,3L,,MeetingRoom
var r = rooms[1]; // 2L
r = rooms["1"]; // 2L
r = rooms["01"]; // undefined
r = rooms[4]; // MeetingRoom
r = rooms[3]; // undefined
```

- not a primitive but an object
- Functions are first-class:
  - can be passed to and returned from other functions
  - can be assigned to variables

```
function makeMultiple(x) {  
  var mult = 2;  
  return mult * x;  
}  
  
// var wrong = mult;      // mult is function-scoped  
var y = makeMultiple(5);  // 10  
var f = makeMultiple;    // function makeMultiple(x) { var m  
var n = f(100);          // 200
```

- `makeMultiple()` refers to the function invocation
- `makeMultiple` refers to the object function

```
var g = function(x){  
    return x + "!";  
}
```

```
g("hey"); // hey!
```

```
g          // function (x){ return x + "!"; }
```

- closure

```
function createFunc() {  
  const x = 20;  
  function f() {  
    return x; // this `x` refers to the local `x` above  
  }  
  return f;  
}  
  
var f1 = createFunc();  
var y = f1(); // 20  
  
var q = (x) => x*3;  
var triple = q(6); // 18
```

```
for (let i = 0; i < rooms.length; i++) { ... }  
for (let k in person) { text += person[k]; }  
for (let j of rooms) { ... }  
rooms.forEach((item) => p(item)); // function p() will be called  
// for each non-empty value of array rooms
```

- Javascript can change the page content (HTML, CSS and log). Use:
  - *innerHTML* to write into an HTML element
  - *document.write()* to write into the entire HTML (may overwrite it)
  - *window.alert()* to write into an alert box
  - *console.log()* to write into the browser console

```
<p id="demo"></p>
```

```
<script>
```

```
document.getElementById("demo").innerHTML = 5 + 6;
```

```
// will write 11 in the paragraph above
```

```
document.write(5 + 6);
```

```
// will append 11 to the document.
```

```
window.alert(5 + 6);
```

```
console.log(5 + 6);
```

```
</script>
```

```
<p id="demo">Will be red</p>
```

```
<script>
```

```
document.getElementById("demo").style = "color:red";
```

```
</script>
```

```
const myElem = document.createElement('span');  
myElem.id = 'bar';  
myElem.style = "color:red";  
myElem.innerHTML = "this is red";  
document.body.appendChild(myElem);
```

## export

- declaration used to export values from a JavaScript module
- adding type="module" to the tag, the runtime interprets the html file as a module
- exported values can be imported in other modules
- a value in an imported binding will change if it changes in the exporting module

```
export let name1, name2/*, ... */; // also var and const
export function functionName() { /* ... */ }
export { variable1 as name1, variable2 as name2, /* ..., */ name3 as name4 }
export default function functionName() { /* ... */ }
export * from "module-name"; // aggregating modules
```

## import

```
// file test.js
const k = 12;
export default k;

// some other file
import m from './test'; // we can use m instead of
                        // k, because k was
                        // default export
console.log(m);         // will log 12
```

- <https://en.wikipedia.org/wiki/JavaScript>
- <https://www.w3schools.com/js/default.asp>
- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language\\_Overview](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Overview)