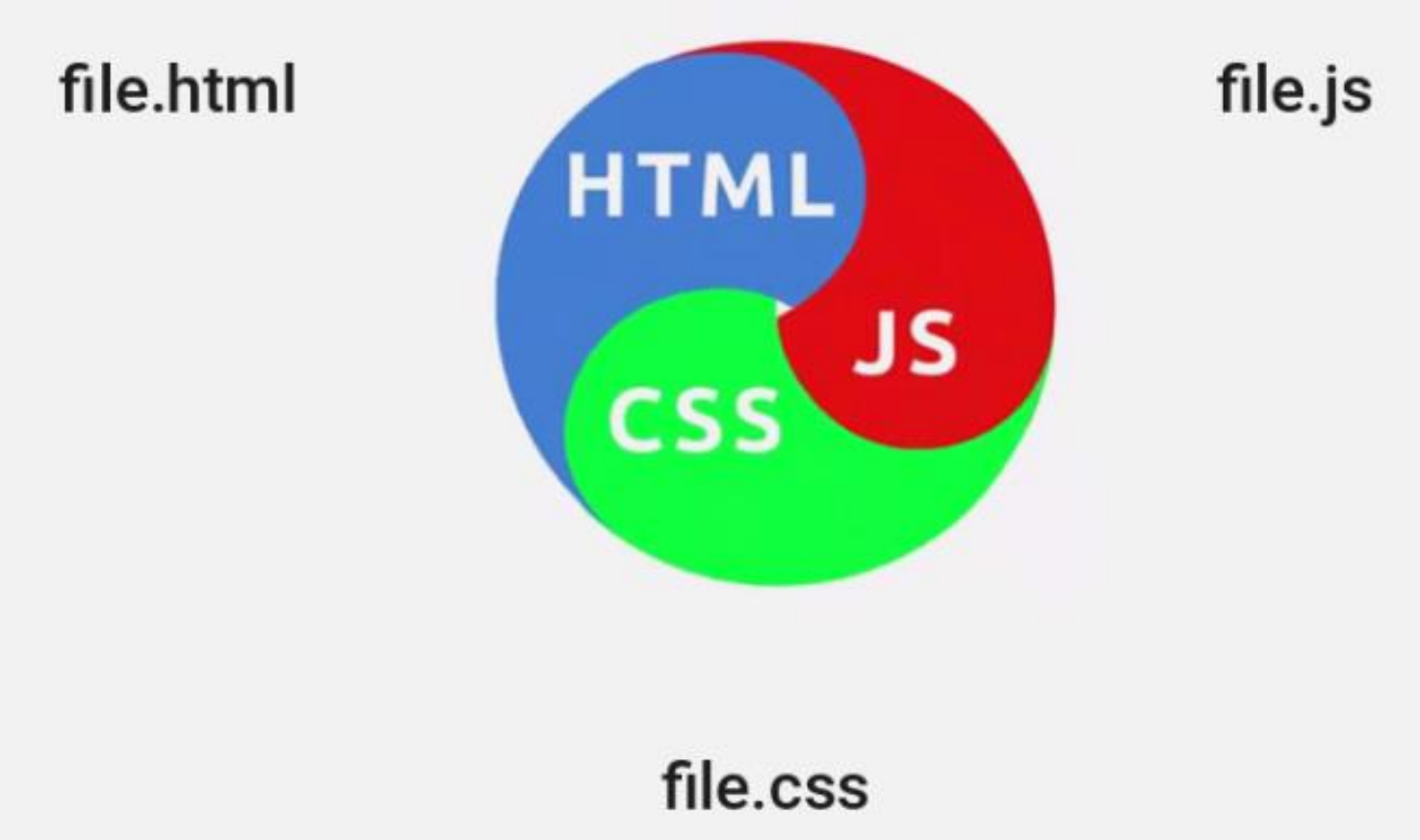
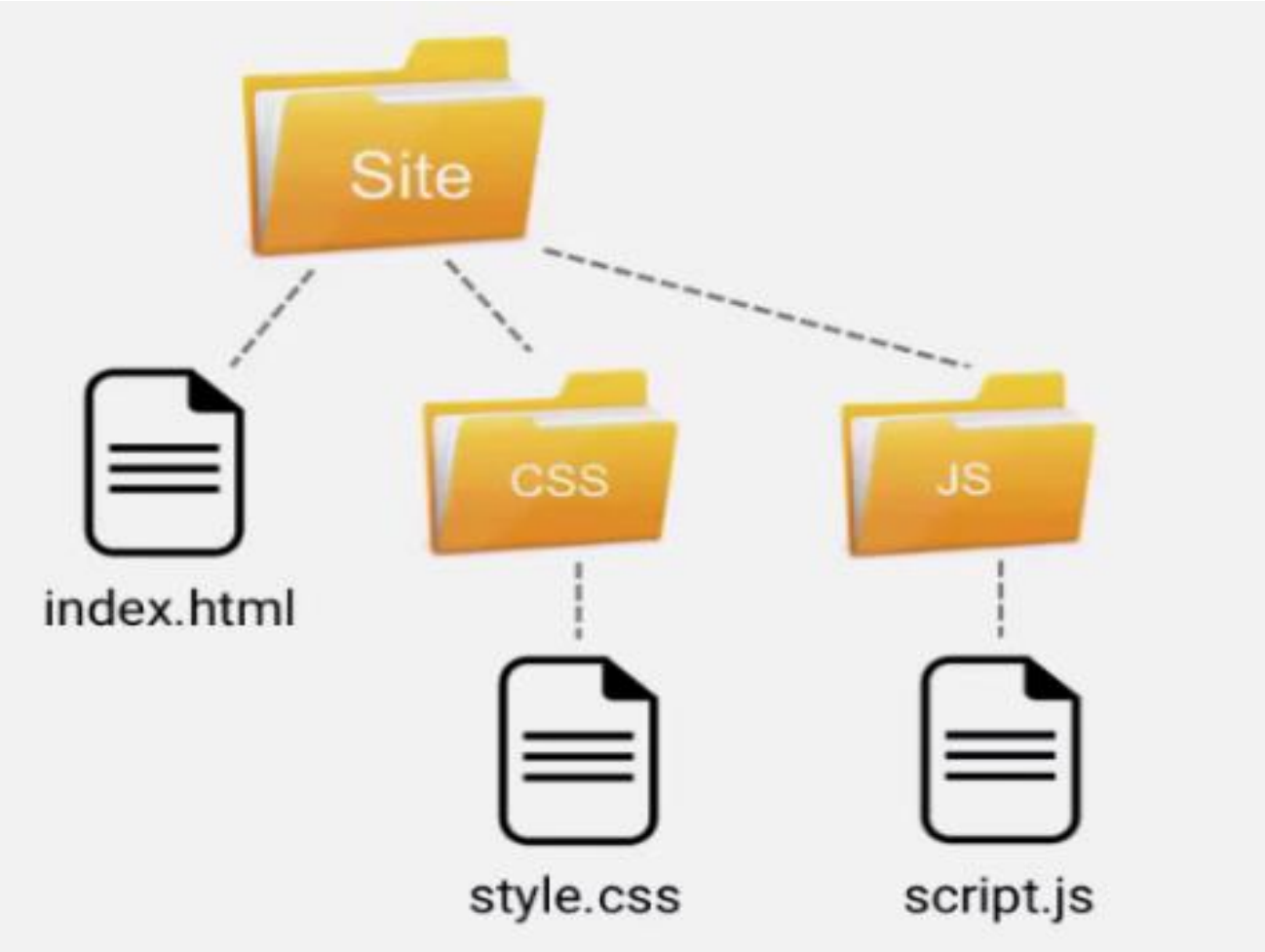

HTML
JS
CSS

JS = JavaScript


file.html
file.js
HTML
JS
CSS
file.css


Site
index.html
CSS
JS
style.css
script.js

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>JS</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <h1>Have a nice day!</h1>

    <script src="path to file"></script>
</body>
</html>
```

script.js

```
alert("Text");
```

script.js

```
alert("Hello, World!")
alert("How are you?")
alert("Have a nice day!")
```

```
alert("Hello, World!") ;
alert("How are you?") ;
alert("Have a nice day!") ;
```

```
alert("Hello!")  alert("How are you?")  alert("Have a nice day!")
```

1 2 3

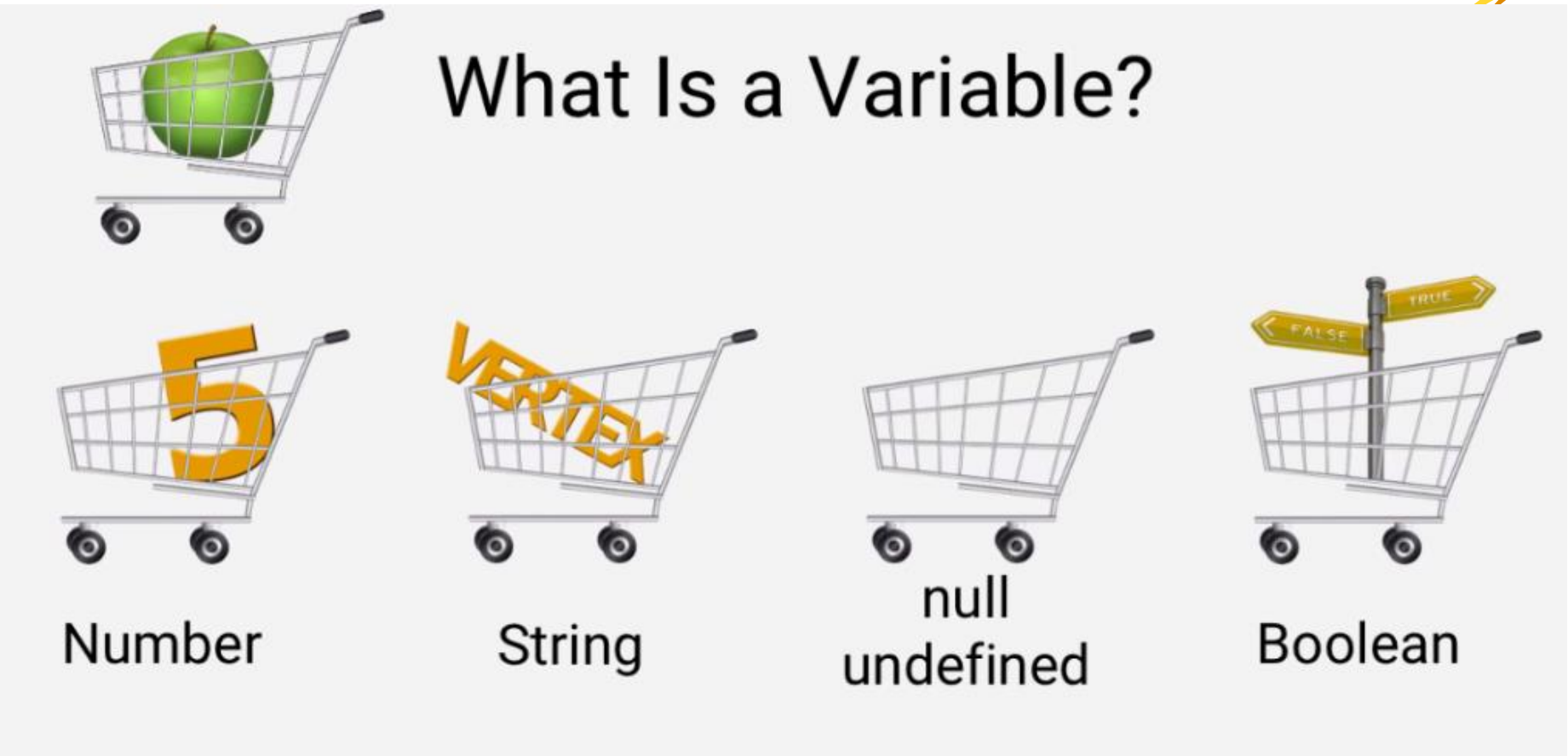
Numbers

, , .

Punctuation

It is as easy as

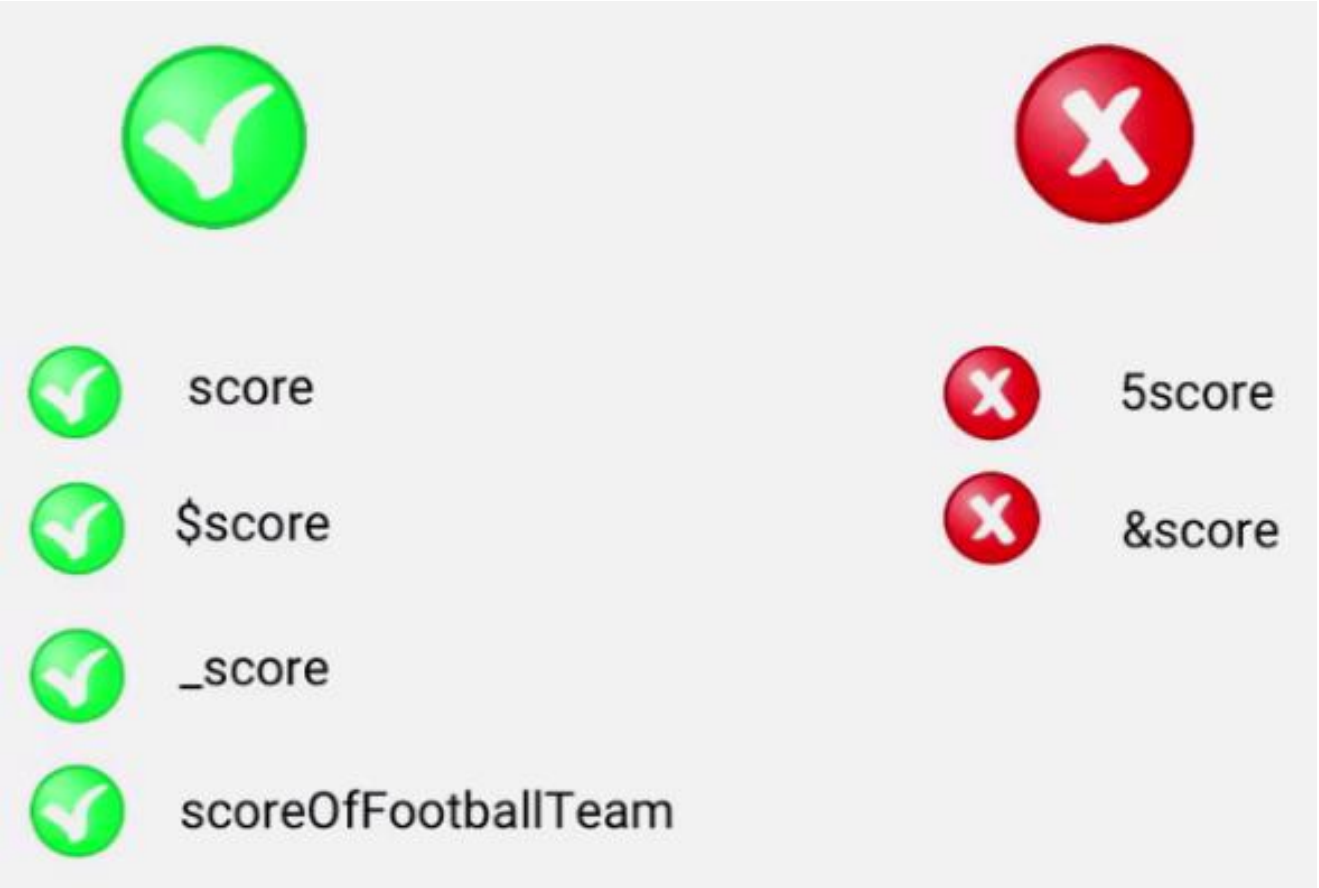
Words


What Is a Variable?
5
VERTEX
TRUE
FALSE
Number
String
null
undefined
Boolean

Dynamic typing

JavaScript is a *loosely typed* and *dynamic* language. Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types:

```
let foo = 42;    // foo is now a number
foo     = 'bar'; // foo is now a string
foo     = true;  // foo is now a boolean
```


score
$score
_score
scoreOfFootballTeam
5score
&score

let

Option 1:

```
let x;
x = 5;
```

Option 2:

```
let x = 5;
```

Operator	Description	Example
+	Addition	3 + 10
-	Subtraction	5 – 2
*	Multiplication	2 * 5
/	Division	10 / 5
%	Remainder	5 % 2


```
let name = "text" or 'text' or `text`
```

```
let name = "Walt"

let name = 'Walt'

let text1 = "If you can dream it, you can do it!"
```

text[index]

text	W	a	l	t		D	i	s	n	e	y
index	0	1	2	3	4	5	6	7	8	9	10

“Walt Disney”[3]

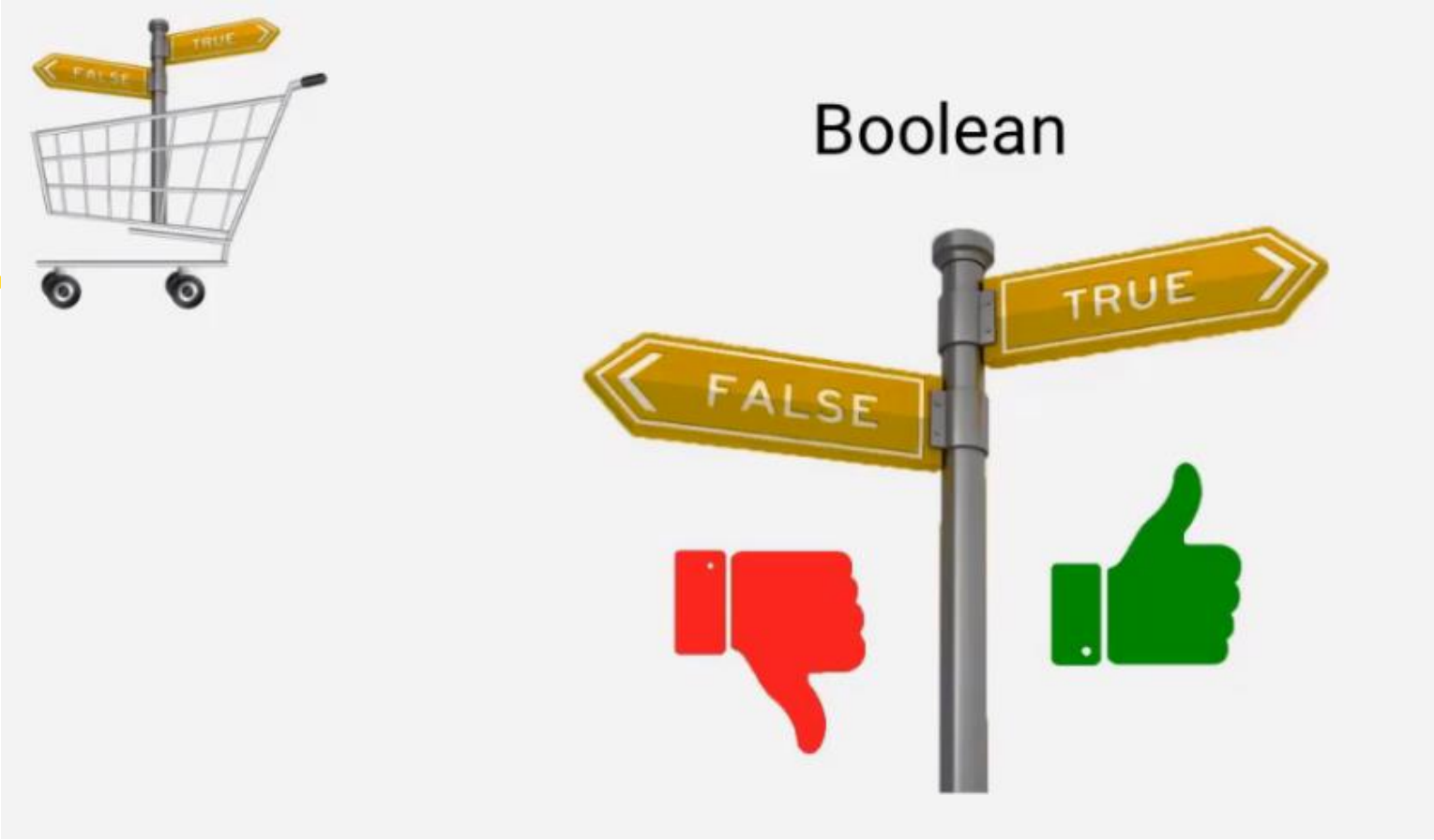
t

let pioneer = “Walt Disney”

pioneer[3]

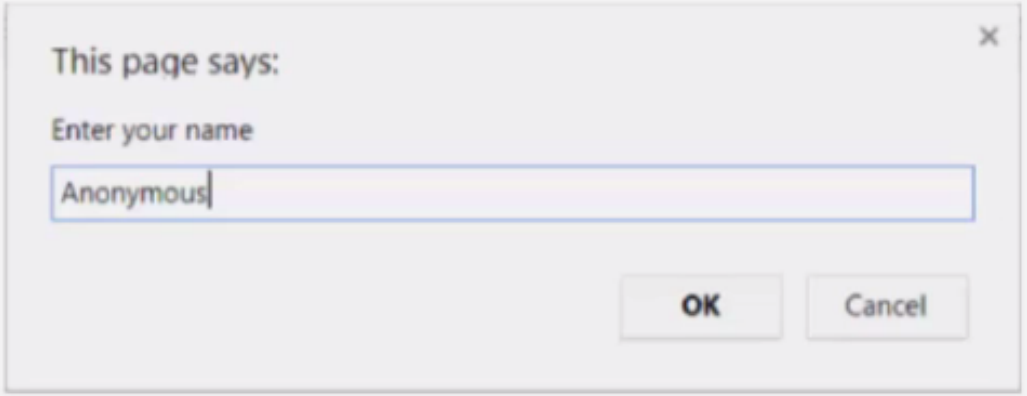
t


undefined and null
let x;
undefined
x = 5;
5
x = null;


Boolean
TRUE
FALSE

confirm(question)

prompt(title, default)



Comparison Operators

Operator	Name	Example	Result
<	Less than	3 < 2	false
<=	Less than or equal	5 <= 5	true
>	Greater than	3 >10	false
>=	Greater than or equal	3 >= 3	true
==	Equality	3 == "3"	true
!=	Inequality	6 != "6"	false
===	Identity / strict equality	3 === "3"	false
!==	Non-identity / strict inequality	6 !== "6"	true

Logical Operators

Operator	Name	Example	Result
&&	AND	(3 < 2) && (3===3)	false
||	OR	(3 < 2) || (3===3)	true
!	NOT	!(3 < 2)	true

Converting to Boolean

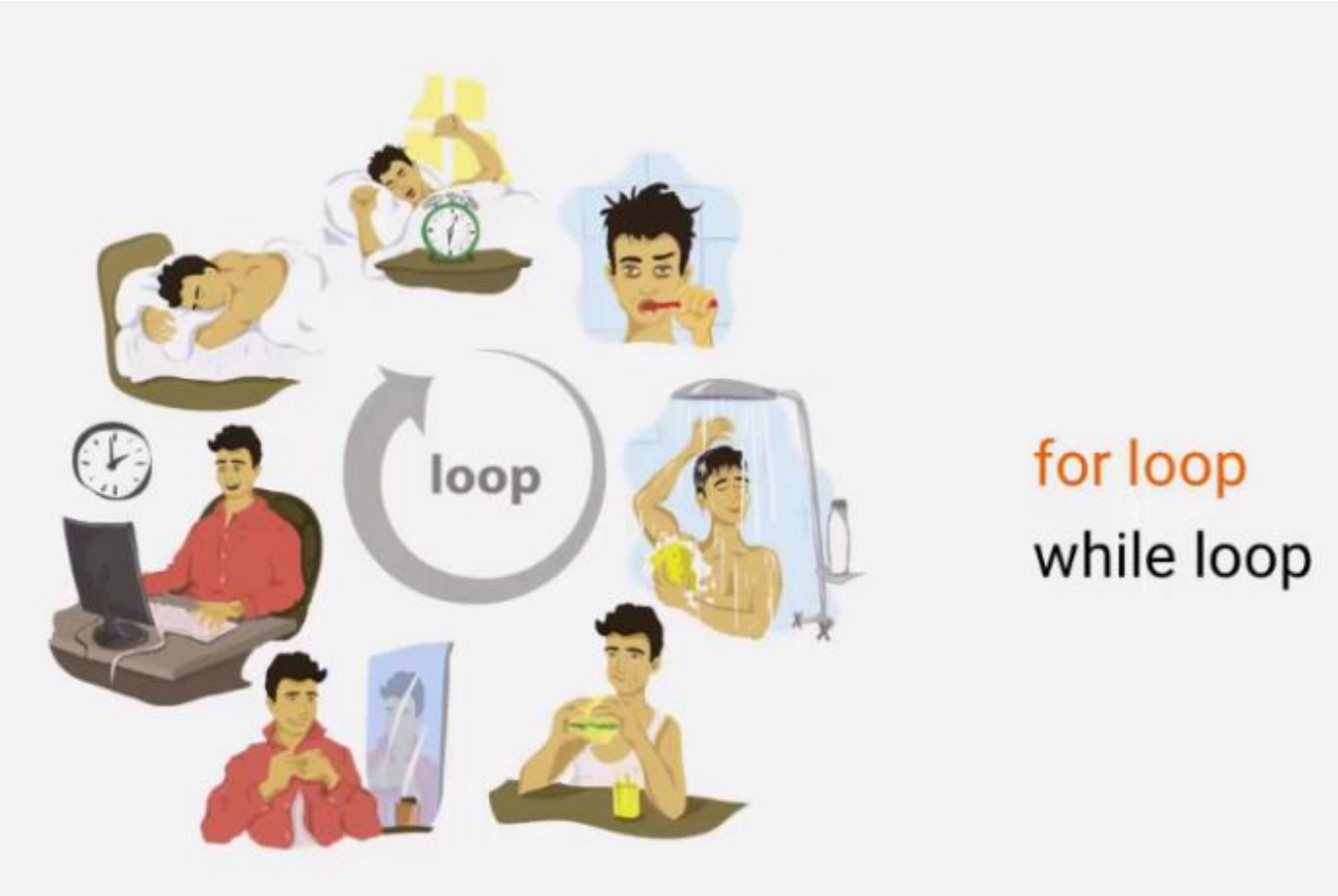
```
let a = "10";        //string

let b = !!a;         //boolean

console.log(b);   =>   true

let c = "";        //string

console.log(!!c);   =>   false
```

Converting to Boolean

```
let a = "10";      //string

let b = !!a;       //boolean

console.log(b);  =>  true

let c = "";        //string

console.log(!!c);  =>  false
```

false	true
false	Everything else
0	
""	
null	
undefined	
NaN	

```
if(condition) {
        block of code to be executed
}

else if(condition) {
        block of code to be executed
}

else {
        block of code to be executed
}
```


loop
for loop
while loop

```
typeof value
```

typeof value

```
let x = “Hello!”;

console.log(typeof x);                    ⇒ string

console.log(typeof 12);                   ⇒ number

console.log(typeof “Have a nice day!”);    ⇒ string
```

What is a function?

1. Functions without arguments
2. Functions with arguments
3. Return

```
function name();


function name(argument1, argument2);
```

```
*
* *
* * *
* * * *
* * * * *
```

1

```
console.log ("*");
console.log ("* *");
console.log ("* * *");
console.log ("* * * *");
console.log ("* * * * *");
```

```
*
* *
* * *
* * * *
* * * * *
```

2

```
console.log ("*");
console.log ("* *");
console.log ("* * *");
console.log ("* * * *");
console.log ("* * * * *");
```

1

```
console.log ("*");
console.log ("* *");
console.log ("* * *");
console.log ("* * * *");
console.log ("* * * * *");
```

2

```
console.log ("*");
console.log ("* *");
console.log ("* * *");
console.log ("* * * *");
console.log ("* * * * *");
```

```
function asterisks() {
   console.log ("*");
   console.log ("* *");
   console.log ("* * *");
   console.log ("* * * *");
   console.log ("* * * * *");
}

asterisks();
asterisks();
```

$x^2 = x * x$

$2^2 = 2 * 2$

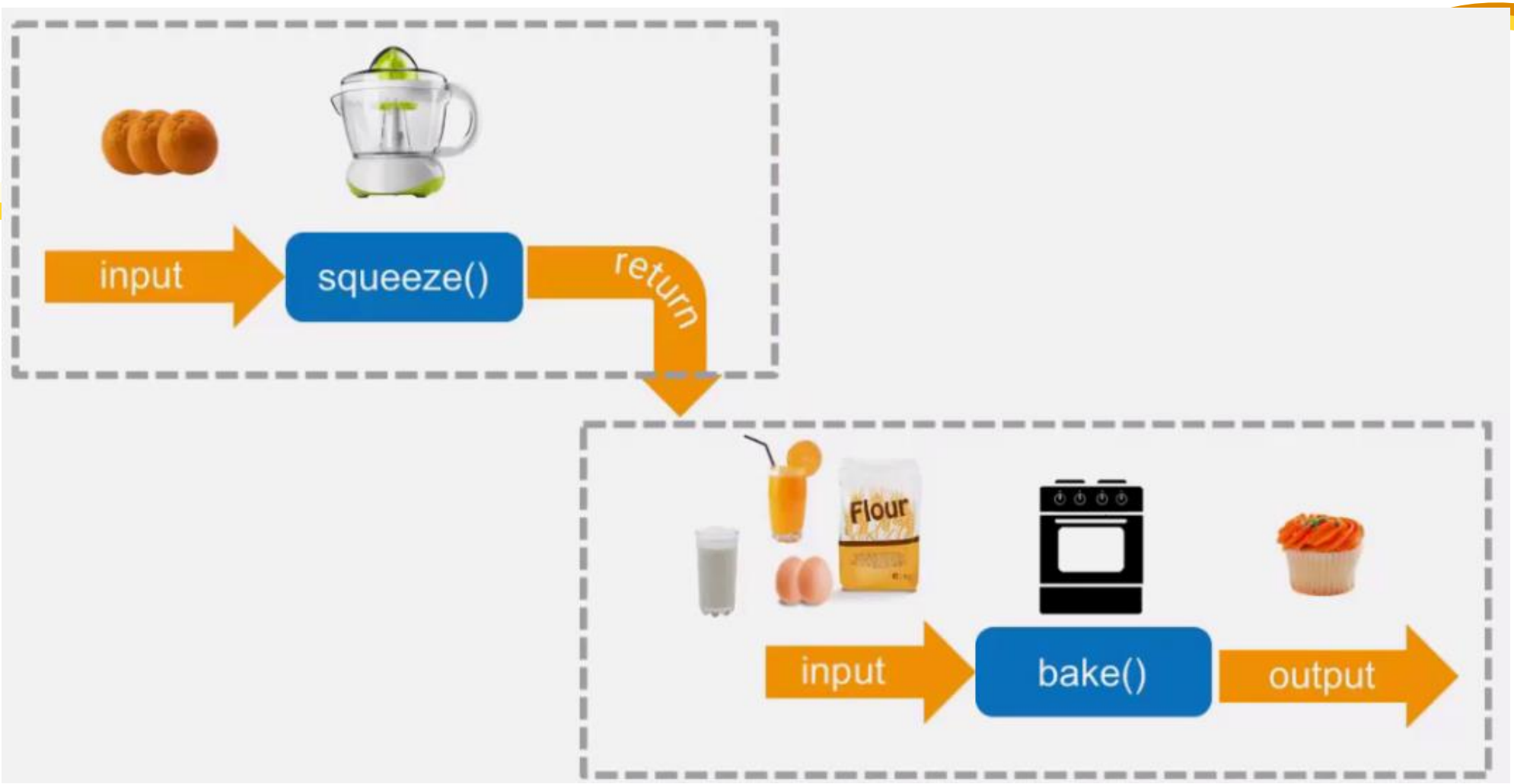
$3^2 = 3 * 3$

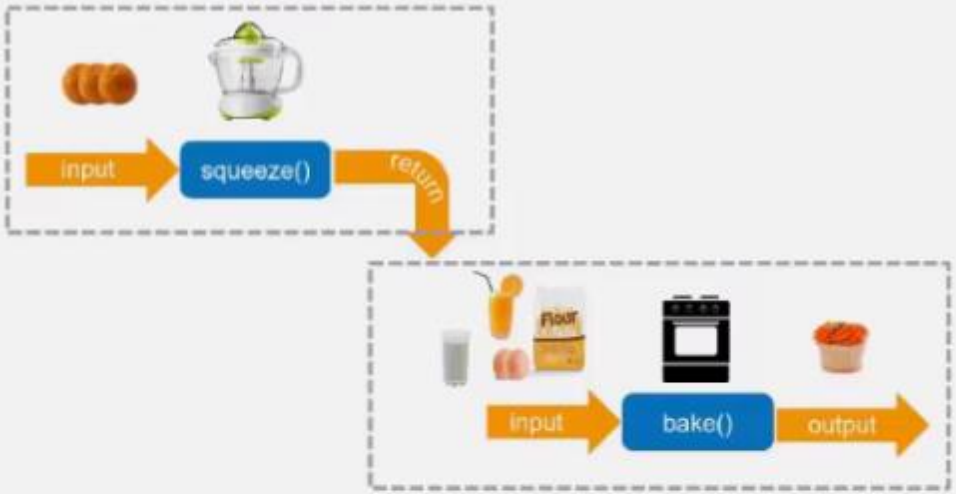
$5^2 = 5 * 5$

```
function square(x){
    let result = x*x;
    console.log (result);
}
```

```
square(2);    ⇒ 4
square(3);    ⇒ 9
square(5);    ⇒ 25
```


```
function squeeze(fruit){
    let juice = fruit + " juice";
    console.log(juice);
}
```


input
squeeze()
return
Flour
input
bake()
output




```
function squeeze(fruit){
  let juice = fruit + " juice";
  console.log(juice);
  return juice;
 }

function bake(ing){
  if(ing.includes('orange juice') && ing.includes('milk') &&
    ing.includes('eggs') && ing.includes('flour')){
      console.log('The cupcake has been baked!');
  }
  else {
    console.log('You don't have all the ingredients');
  }
}

let juice = squeeze('orange');
bake([juice, 'eggs', 'milk', 'flour' ]);
```

⇒ orange juice
The cupcake has been baked!

Callback function

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

Here is a quick example:

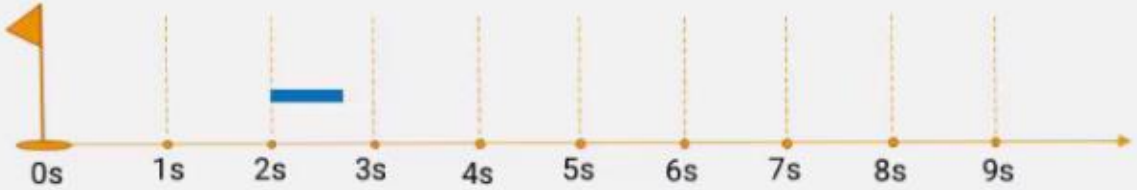
```
function greeting(name) {
  alert('Hello ' + name);
}

function processUserInput(callback) {
  var name = prompt('Please enter your name.');
  callback(name);
}

processUserInput(greeting);
```

setInterval() vs setTimeout()

2s

```
setInterval(function(){...}, 2000);
```


```
setTimeout(function(){...}, 2000);
```



- function

setInterval() vs setTimeout()

2s

```
setInterval(function(){...}, 2000);
```


```
let counter = 0;
let interval = setInterval(function(){
   counter++;
   if(counter===10){
      clearInterval(interval);
   }
}, 2000);
```

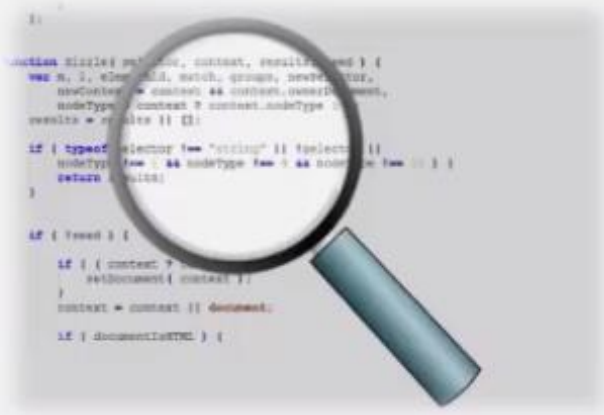
Scope

```
<script>
    let x = 2;
    let text = 'Luck';
    function printX () {
        let x = 10;
        console.log(x);
    }
    function sayText () {
        let text ='Hello';
        console.log(text);
    }
</script>
```

Global Scope

Local Scope

Local Scope



Scope

```
<script>
  let x = 2;
  function printX () {
    let x = 10;
    for(let x = 0; x <= 2; x++) {
      console.log(x);        ⇒ 0
    }                          1
                               2
    console.log(x);          ⇒ 10
  }
  printX();
  console.log(x);            ⇒ 2
</script>
```

Global Scope

Local Scope

Local Scope

- The `argument` object is created, containing all the arguments that were passed into the function.
- Code is scanned for **function declarations**: for each function, a property is created in the Variable Object, **pointing to the function**.
- Code is scanned for **variable declarations**: for each variable, a property is created in the Variable Object, and set to `undefined`.

HOISTING


```
// functions
calculateAge(1965);

function calculateAge(year) {
    console.log(2016 - year);
}


// retirement(1956);
var retirement = function(year) {
    console.log(65 - (2016 - year));
}



// variables

console.log(age);
var age = 23;
```


```
let year = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];

console.log(year[2] );   ⇒   "Mar"        year[2] = "March"
console.log(year[6] );   ⇒   "Jul"        year[6] = "July"
console.log(year[11] );  ⇒   "Dec"        year[11] = "December"
```


```
let months = ["Feb", "Mar", "Apr", "May"];

.push(element);

months.push("Jun");
```


```
let months = ["Feb", "Mar", "Apr", "May"];

.unshift(element);

months.unshift("Jan");
```


```
let months = ["Feb", "Mar", "Apr", "May", "Jun"];


.shift();

months.shift();
```


```
let months = ["Feb", "Mar", "Apr", "May", "Jun"];

.pop();

months.pop();
```


```
let months = ["Feb", "Mar", "Apr", "May", "Jun"];
```

```
.slice(from, to);
```

```
months.slice(1, 4);
```

```
.slice(from);
```

```
months.slice(1);
```

```
.slice();
```

```javascript
let months = [3,4,5,6];

// add
months.unshift(2); // unshift(elelment) = add(0, element)
// --> expected: 2 3 4 5 6

months.push(7); // push(element) = add(end, element);
// --> expected: 2 3 4 5 6 7

months.splice(2, 0, 99); // splice(index, 0, element) = add(index, element)
// --> expected: 2 3 99 4 5 6 7

// remove
months.shift(); // shift = remove(0)
// --> expected: 3 99 4 5 6 7

months.pop(); // pop = remove(end)
// --> expected: 3 99 4 5 6

months.splice(2, 2); // splice(index, count) = remove(index, elementCountsRemoved)
// --> expected: 3 99 6

// replace
months.splice(0, 3, 1, 2, 3); // replace 3 elements from index 0 by 1 2 3
// --> expected: 1 2 3

months.splice(2, 1, 4);
// --> expected: 1 2 4

alert(months);
```

```javascript
Array.prototype.insert = function(idx, element) {
    this.splice(idx, 0, element);
}
```

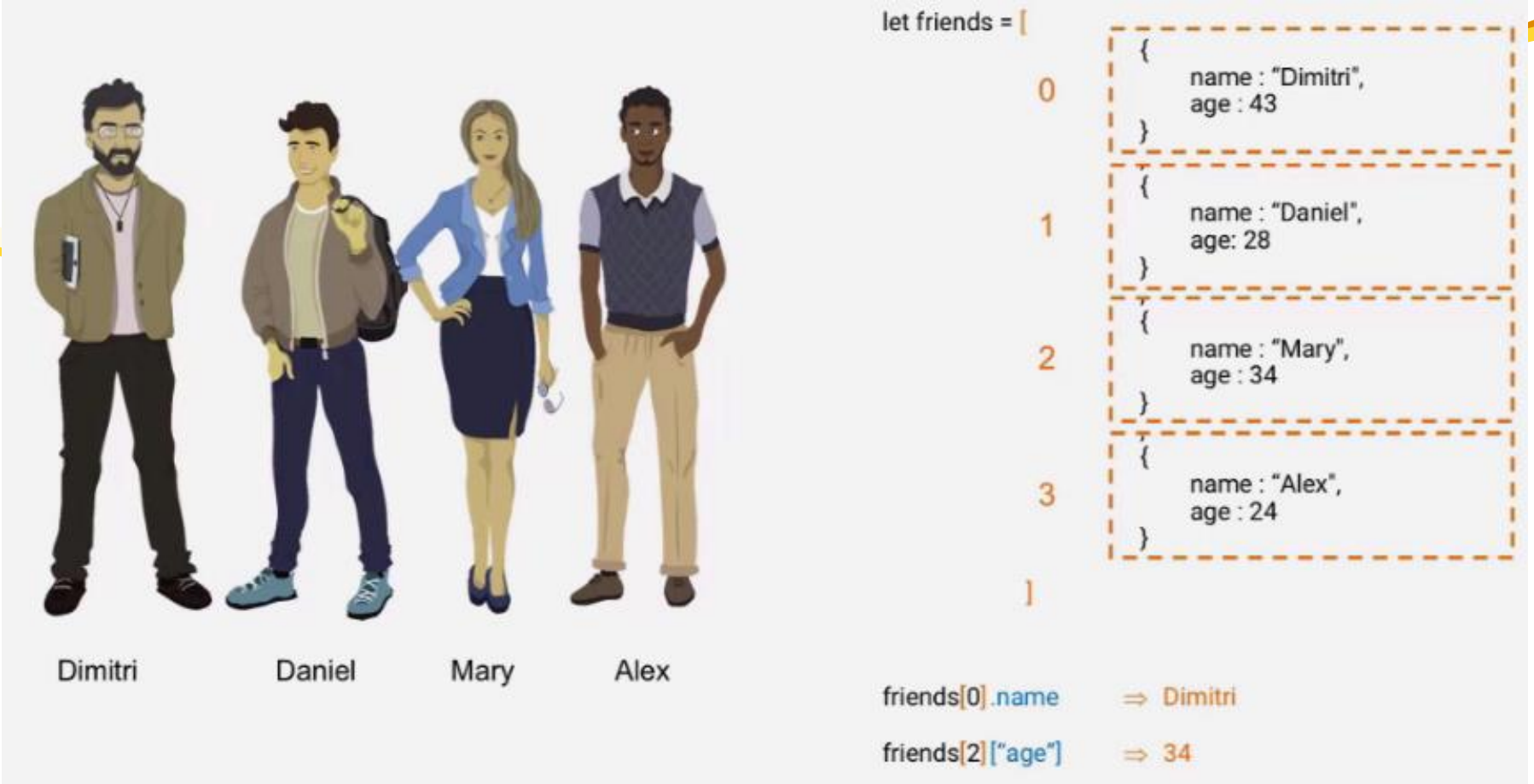
```
let person = [ "Daniel", "Jones", 28, "male"]
```

```
let person = {
                 name: "Daniel",
                 surname: "Jones",
                 age: 28,
                 sex: "male"
             }
```

property value

```
console.log( person.age );          ⇒ 28
// or
console.log( person["age"]);        ⇒ 28
```

```
person.age++;
// or
person["surname"] = "Whitman"
```


Dimitri
Daniel
Mary
Alex
let friends = [
0
{
name : "Dimitri",
age : 43
}
1
{
name : "Daniel",
age: 28
}
2
{
name : "Mary",
age : 34
}
3
{
name : "Alex",
age : 24
}
]
friends[0].name ⇒ Dimitri
friends[2]["age"] ⇒ 34


```
let friends = [
    {
        name : "Dimitri",
        age : 43,
        parents: [
            {
                name: "Tom",
                age: 67
            }
            ,
            {
                name: "Kate",
                age: 64
            }
        ]
    }
    ,
    {
        name : "Daniel",
        age: 28,

    }
]
```

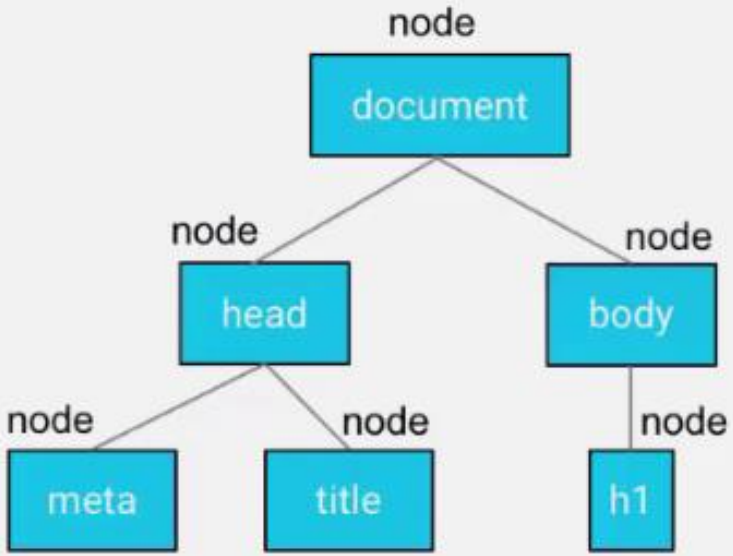
DOM

Document Object Model

A Web page is a document

HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>JS</title>
  </head>
  <body>
    <h1 id="page-heading" class="red">Hello</h1>
  </body>
</html>
```




```
let tag = {
        tagName: "h1",
        id: "page-heading",
        className: "red",
        textContent: "Hello"
        }
```

Nodes:

1. Element nodes
2. Text nodes

Introduction to the DOM

The **Document Object Model** (**DOM**) is the data representation of the objects that comprise the structure and content of a document on the web. In this guide, we'll briefly introduce the DOM. We'll look at how the DOM represents an HTML or XML document in memory and how you use APIs to create web content and applications.

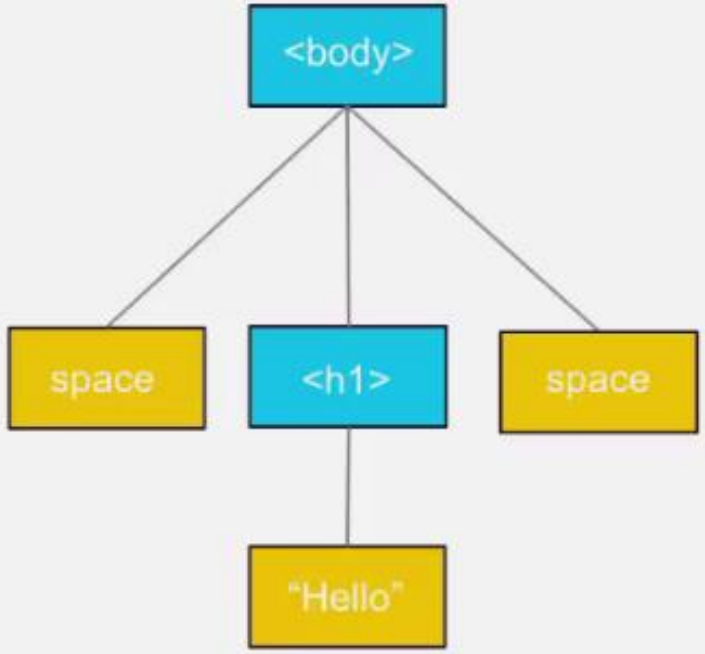
What is the DOM?

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects. That way, programming languages can connect to the page.

DOM and JavaScript

The short example above, like nearly all of the examples in this reference, is JavaScript. That is to say, it's *written* in JavaScript, but it *uses* the DOM to access the document and its elements. The DOM is not a programming language, but without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents, XML documents, and their component parts (e.g. elements). Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of the document object model for that document, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.

```
<body>
    <h1 id="page-heading"  class="red">Hello</h1>
</body>
```




```
let tag = {
        tagName: "h1",
        id: "page-heading",
        className: "red",
        textContent: "Hello"
        }
```

`window` object in the Browser

The `window` object is the Global Object in the Browser. Any Global Variables or Functions can be accessed as *properties* of the `window` object.

Access Global Variables

```
var foo = "foobar";
foo === window.foo; // Returns: true
```

After defining a Global Variable `foo`, we can access its value directly from the `window` object, by using the variable name `foo` as a property name of the Global Object `window.foo`.

Explanation:

The global variable `foo` was stored in the `window` object, like this:

```
foo: "foobar"
```

JS Selectors

```
1  .getElementsByName("name")
2  .getElementsByTagName("tag")
3  .getElementsByClassName("class")
4  .getElementById("id")
5  .querySelector("cssSelector")
6  .querySelectorAll("cssSelector")
```

1

getElementsByName()

```
<body>
    <h1 class="red-text">Greeting</h1>
    <p class="red-text" id="main">Hello!</p>
    <p class="green-text" name="wish">Have a nice day!</p>
</body>
```

```
let name = document.getElementsByName("wish")
console.dir(name);
```

2

getElementsByTagName()

```
<body>
    <h1 class="red-text">Greeting</h1>
    <p class="red-text" id="main">Hello!</p>
    <p class="green-text" name="wish">Have a nice day!</p>
</body>
```

```
let tag = document.getElementsByTagName("p")

console.dir(tag);
```

3

getElementsByClassName()

```
<body>
    <h1 class="red-text">Greeting</h1>
    <p class="red-text" id="main">Hello!</p>
    <p class="green-text" name="wish">Have a nice day!</p>
</body>
```

```
let className = document.getElementsByClassName("red-text")

console.dir(className);
```

4

getElementById()

```
<body>
    <h1 class="red-text">Greeting</h1>
    <p class="red-text" id="main">Hello!</p>
    <p class="green-text" name="wish">Have a nice day!</p>
</body>
```

```
let id = document.getElementById("main")
console.dir(id);
```

5

querySelector()

```
<body>
    <h1 class="red-text">Greeting</h1>
    <p class="red-text" id="main">Hello!</p>
    <p class="green-text" name="wish">Have a nice day!</p>
</body>
```

```
let query = document.querySelector(".red-text")
console.dir(querty);
```

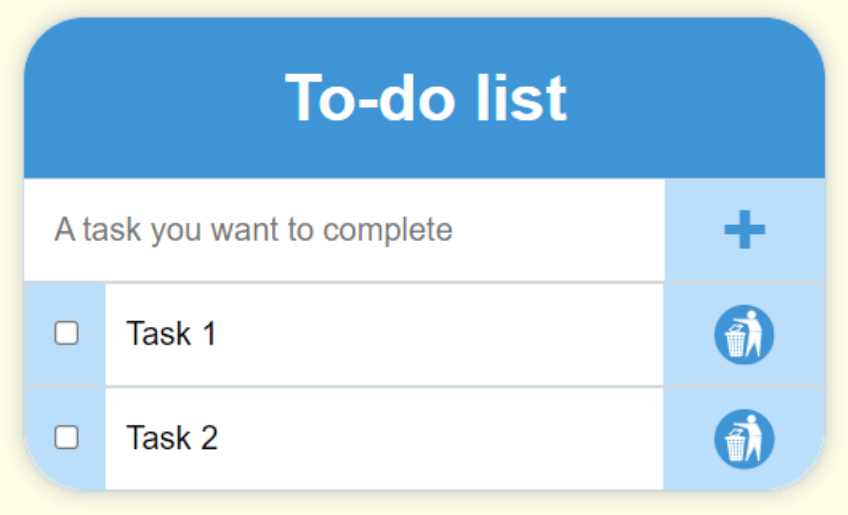
6

querySelectorAll()

```
<body>
    <h1 class="red-text">Greeting</h1>
    <p class="red-text" id="main">Hello!</p>
    <p class="green-text" name="wish">Have a nice day!</p>
</body>
```

```
let query = document.querySelectorAll(".red-text")
console.dir(querty);
```

```
1   document.createElement('tag name');
2   .appendChild(element);
3   .insertBefore(element, anotherElement);
4   .remove()
```



HTML

```
<h2>To-do list</h2>
<ul id="list">
    <li>task 1<li>
    <li>task 2<li>
<ul>
```

JS

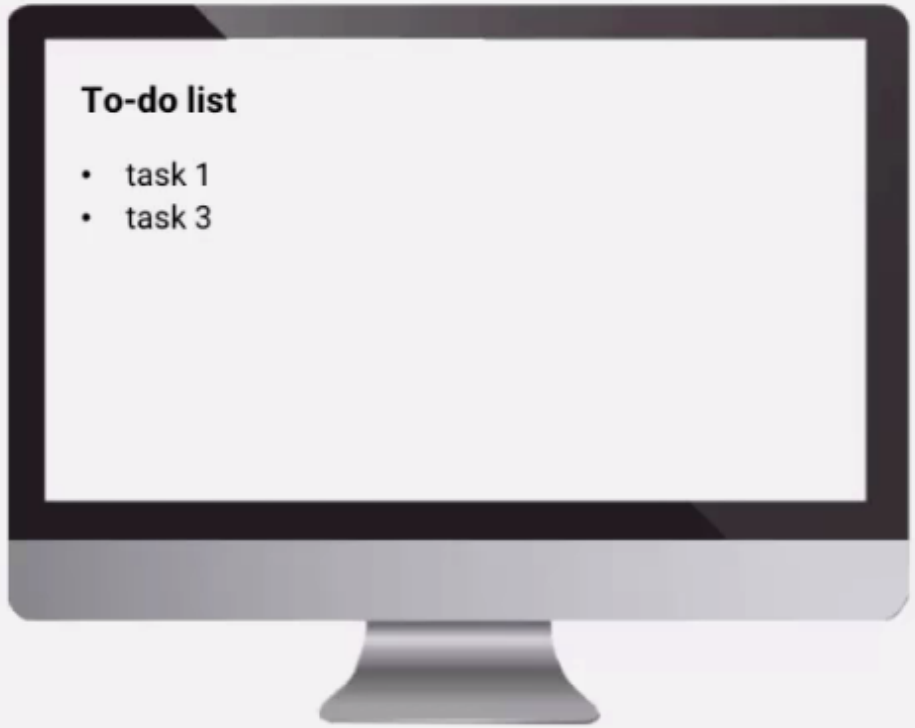
```
let task = document.createElement('li');
task.textContent = 'task 3';
let list = document.getElementById('list');
list.appendChild(task);
```

HTML

```
<h2>To-do list</h2>
<ul id="list">
    <li>task 1<li>
    <li>task 2<li>
<ul>
```

JS

```
let task = document.createElement('li');
task.textContent = 'task 3';
let list = document.getElementById('list');
let firstElement = list.children[0];
list.insertBefore(task, firstElement);
```



HTML

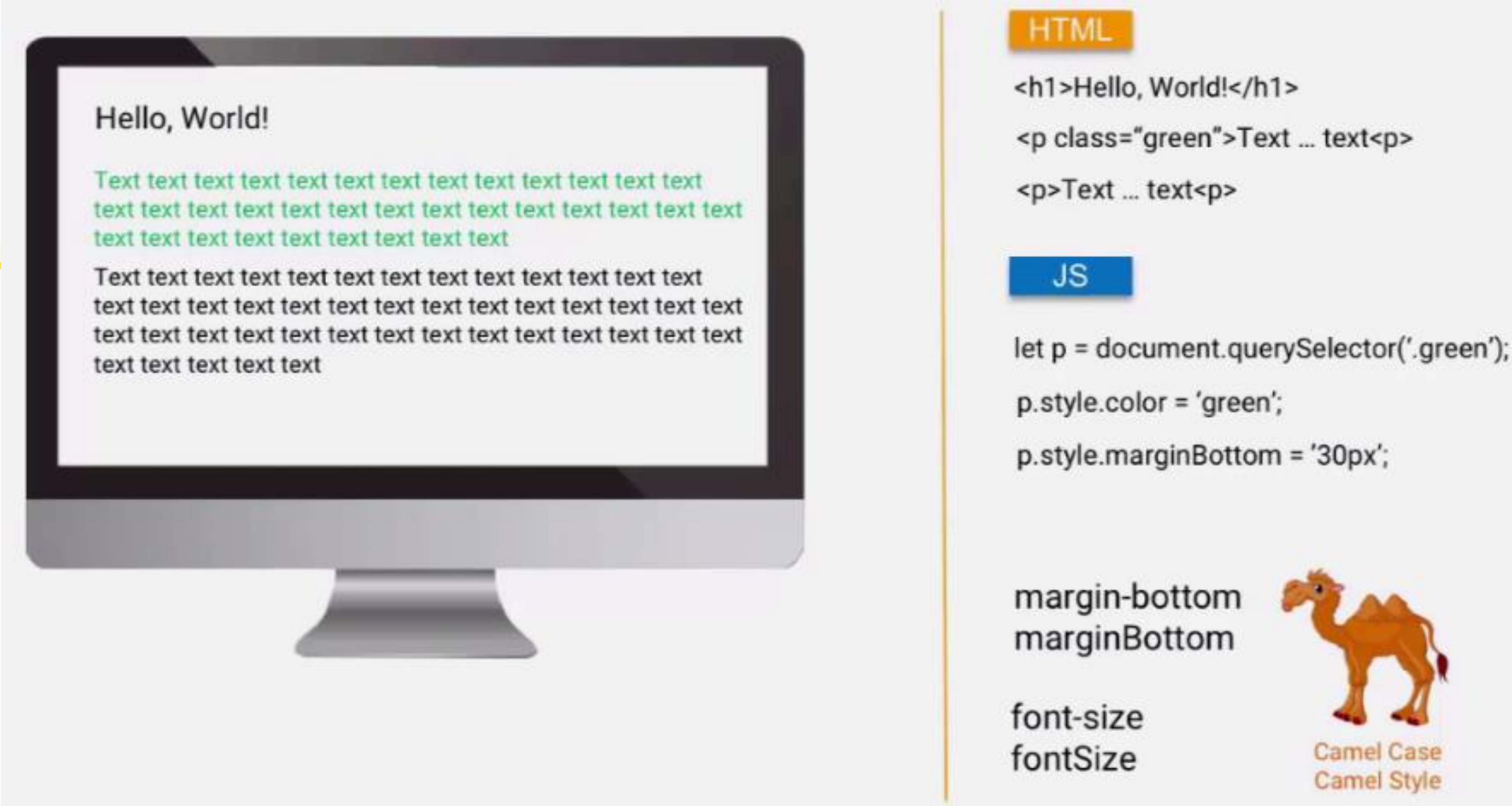
```
<h2>To-do list</h2>
<ul id="list">
    <li>task 1<li>
    <li>task 2<li>
    <li>task 3<li>
<ul>
```

JS

```
let list = document.getElementById('list');
let element = list.children[1];
element.remove();
```

Changing Element's Style

```
elem.style.property = "value";

elem.style.width = "200px";
elem.style.color = "#3300ff";
elem.style.display = "flex";
```



Hello, World!

Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text

Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text

HTML

```
<h1>Hello, World!</h1>
<p class="green">Text ... text<p>
<p>Text ... text<p>
```

JS

```
let p = document.querySelector('.green');
p.style.color = 'green';
p.style.marginBottom = '30px';
```

margin-bottom
marginBottom

font-size
fontSize

Camel Case
Camel Style

Basic Events

click
contextmenu
mouseover
mousedown
mouseup
keypress
submit

HTML

```
<button class="btn">Click me!</button>
```

JS

```
let btn = document.querySelector('.btn');

btn.onclick = function(){
    console.log('You have single-clicked!');
}

btn.onmouseover = function(){
    console.log('Your pointer is over the button!');
}
```

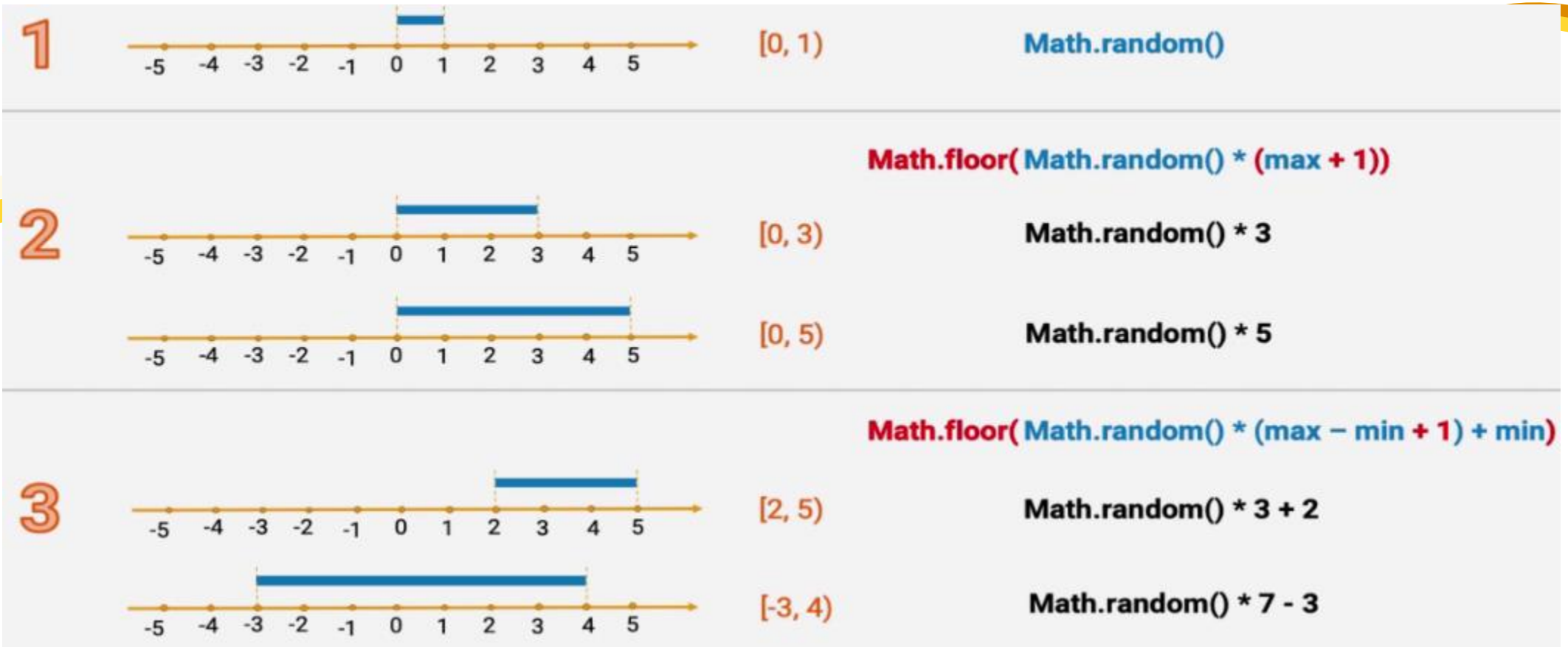
HTML

```
<button class="btn">Click me!</button>
```

JS

```
let btn = document.querySelector('.btn');

btn.addEventListener('click', function(){
    console.log('Hello!')
})

btn.addEventListener('click', function(){
    console.log('Bye!');
})
```

Random Number Generation


1
-5 -4 -3 -2 -1 0 1 2 3 4 5
[0, 1)
Math.random()
Math.floor(Math.random() * (max + 1))
2
-5 -4 -3 -2 -1 0 1 2 3 4 5
[0, 3)
Math.random() * 3
-5 -4 -3 -2 -1 0 1 2 3 4 5
[0, 5)
Math.random() * 5
Math.floor(Math.random() * (max – min + 1) + min)
3
-5 -4 -3 -2 -1 0 1 2 3 4 5
[2, 5)
Math.random() * 3 + 2
-5 -4 -3 -2 -1 0 1 2 3 4 5
[-3, 4)
Math.random() * 7 - 3

Technology

Consulting

5432 Any Street West, Townsville, State 54321

format code: Edit > Line > Reindent

copy line: Ctrl Shift D

delete row: Ctrl X