

Generadores de texto Lorem Ipsum → ver las opciones disponibles.

Font-size: Si se especifica en px no se ve afectado cuando se cambian las preferencias del explorador.

Dynamic font-size: $100\% = 16 \text{ px}$

 $1em = 16 \text{ px}$

Zoom de la página web → no importa si el tamaño de la fuente es estático o dinámico. \rightarrow el tamaño se hereda del padre. **OJO!**
 $\%$ →

rem → se refiere al root em y no hereda de los padres.
 ↳ recomendación de Angelia Yu por ser lo más adaptable y menos propenso a errores.

$16 \text{ px} = 1 \text{ rem}$
 $48 \text{ px} = 3 \text{ rem}$

Regla de tres simple para tamaño de fuentes diferentes a 1 rem ($= 16 \text{ px}$)

Propiedades de fuentes: color, font-weight, line-height

Propiedad **text-align** → tener en cuenta que CSS tiene en cuenta la propiedad más específica.

float property

clear property ← anti float

⚠ Usar float solo cuando es realmente necesario ⚡

Usarlo cuando se quiere poner texto alrededor de un elemento.

¡No usarlo para posicionar elementos! Por eso está la propiedad **position**.

```
@media <type> <feature>
@media screen (min-width: 900px) {
    // change something
}
```

```
@media (max-width: 1250px) {
```

CSS se aplica si el viewport del explorador es igual o más ancho que 1250 px.

```
}
```

Otras features: min-height, max-height.

```
@media (30cm <= width <= 50cm) {
    /* - */
}
```

Java Script

Java → lenguaje compilado (como C/C++ y Swift)

JavaScript → lenguaje interpretado. (como Python y Ruby)

- alert("Hello"); → muestra un cuadro de diálogo con el texto "Hello".
- Chrome → console → para probar líneas de código.
- Sources → Snippets. → para probar snippets.

Ver video

folio del curso

Tipos de datos: string, numbers, boolean.

- typeof (argumento) → devuelve el tipo de dato.
- Variables.

- prompt(texto); → pide ingresar datos.

- var myName = "Guillermo"; → declara una variable.

- myName = "Ariel"; → cambia el valor de la variable.

- var yourName = prompt ("What's your name?");
- alert ("My name is " + myName + ", welcome " + yourName);

Convención de nombres de variables.

Ctrl + k → limpia la consola

click derecho.

Click sostenido en el botón de actualización del explorador permite varias operaciones, Hard Reload entre ellas.

- Dar nombres significativos a las variables.

- Pueden contener: letras, números, \$, -

- Usar Camel Case.

Strings: se concatenan usando +

⚠ // para introducir comentarios, una linea de comentarios.

- word.length; → devuelve la cantidad de caracteres.

- "holá".length;

- myName.length; null.length; → error.

⚠ /* para introducir varias

líneas de comentarios */

⚠ Ctrl + / → comenta las líneas seleccionadas.

cadena.slice(a, b);

- var nombre = "Guillermo";

- nombre.slice(0, 6); → "Guille" me quedo con 6-0 caracteres.

word.toUpperCase();

nombre.toUpperCase(); → "GUILLERMO"

nombre.toLowerCase(); → "guillermo"

Operaciones con números

resto.

Operadores: +, -, *, /; % (modulo)

even number \rightarrow par } number % 2 = 0 \Rightarrow even
 odd number \rightarrow impar } number % 2 = 1 \Rightarrow odd

$$\text{humanAge} = (\text{dogAge} - 2) * 4 + 21$$

$x++$; equivale a $x = x + 1$; incremento

$x--$; equivale a $x = x - 1$; decremento.

$x += 2$; equivale a $x = x + 2$;

$$\begin{aligned} + &= \left\{ \begin{array}{l} x = x + y \\ x = x - y \\ x = x * y \\ x = x / y \end{array} \right. \\ - &= \\ * &= \\ / &= \end{aligned}$$

operadores equivalentes a

Funciones.

```
function nombreDeFuncion () {      }      // crear la función.
// instrucciones
}
```

nombreDeFuncion (); \leftarrow llama a la función.

funciones con parámetros y argumentos.

```
function nombreDeFuncion (parametro) {      }      // crear la función
// instrucciones que usan el parámetro.
}
```

nombreDeFuncion (argumento); \leftarrow llama a la función.

Solidar o valores devueltos por una función.

```
function nombreDeFuncion (parametro) {
```

// instrucciones

return valor;

}

valor = nombreDeFuncion (argumento);

Números aleatorios.

Var $n = \text{Math.random}();$ → genera un número aleatorio entre 0 y 0.999... 16 lugares decimales.

Para generar números aleatorios entre 1 y 6:

$n = n * 6;$

$n = \text{Math.floor}(n);$

¡Buscar en MDN Web Doc!

Estructura de control: If - Else.

```
if (condición) { // ejecutar esto si es verdadero }
else { // ejecutar esto si falso }
if (track === "clear") {
  goStraight();
} else {
  turnRight();
}
```

Operadores de comparación

$==$ igual a

\neq ~~no~~ distinto a

$>$

$<$

\geq

\leq

$==$ verifica igualdad aún del tipo de datos. $1 == "1" \rightarrow \text{falso}$

$=$ verifica igualdad, pero no del tipo de datos. $1 = "1" \rightarrow \text{true}$

&& AND

|| OR

! NOT

```
if (loveScore > 70) {  
    alert ("Mucho amor");  
}
```

```
if (loveScore > 30 && loveScore <= 70) {  
    alert ("Mejor amor");  
}
```

```
else {  
    alert ("Poco o nada");  
}
```

Arrays 0 1
var eggs = [0, 0, 0, 0];

var myEgg = eggs[1];

eggs.length; \Rightarrow 4

eggs.includes(0); \rightarrow True

eggs.push(0); \rightarrow agrega un elemento al final.

eggs.pop(); \rightarrow elimina el último elemento.

Condicional : if (cond) { código } else if (cond) { código }
 else { código }

While loop

```
while (something is true) {  
    // do something  
}
```

For loops, start end change.
 for (ⁱ⁼⁰; i < 2; i++) {
 // do something
 }
 falta ver ; es opcional?

DOM (Document Object Model)

- inline scripts.

```
<body onload="alert('Hello')">  

</body>
```

- internal javascript.

```
<script type="text/javascript">  

  alert("Hello");  

</script>
```

} dentro de body

- external javascript

```
<script src="index.js" charset="utf-8"></script>
```

Nombre el archivo index.js.

La posición dentro del script dentro del body es muy importante.

La mejor práctica es colocar el código JS al final, justo antes de </body>.

DOM → estructura de árbol. que contiene objetos.

document; → obre la estructura en la consola

document.firstElementChild;

document.firstElementChild.lastElementChild;

var elemento = document.firstElementChild;

elemento. innerHTML = "Nuevo valor";

elemento.style.color = nuevo color;

document.querySelector("input").click();

objetos → propiedades
métodos.

document.getElementsByTagName("nombre de etiqueta"); ← devuelve

document.getElementsByClassName("nombre de clase"); un array.

document.getElementById(id);

document.querySelector("*");

*: etiquetas (por ej: h1)

*: clase (por ej: .my-class)

*: id (por ej: #my-id)

*: se puede combinar para ser más específico (por ej: .my-class h1)

En caso de que querySelector encuentre más de un elemento que satisface la condición, solo trae el primero.

document.querySelectorAll("selector"); → trae todo en un array.

Query Selector se usa más que getElement.

HTML DOM Style Object en w3schools.com están todos los propiedades.

En principio hay una equivalencia CSS-JS del tipo:

font-size → fontSize

background-color → backgroundColor.

Los valores de propiedades en JS son en string.

document.querySelector("xxx").classList;

document.querySelector("xxx").classList.add("nueva clase");

~~10/01 - 24/01 - 31/01~~

document.querySelector("xxx").classList.remove("clase a remover").
Conviene definir las clases en CSS y agregarlas o quitarlas con javascript.

```
document.querySelector("xxx").classList.toggle("clase a alternar");
document.getElementById("id").textContent = "texto";
```

Difiere de innerHTML en que uno trae texto mientras que el otro trae todo lo que se encuentra dentro de las etiquetas seleccionadas.

```
document.querySelector("hf").innerHTML = "<em>Good Bye</em>";
```

Manipulación de atributos.

```
document.querySelector("a").getAttribute("href");
```

```
document.querySelector("a").setAttribute("href", "nuevo atributo");
```

```
[var numberOfButtons = document.querySelectorAll(".drum").length;
for (var i=0; i<numberOfButtons; i++) {
  document.querySelectorAll(".drum")[i].addEventListener("click",
    function() {alert("I got clicked!");});]
```

Higher order functions. → pasando funciones como argumentos.

```
function add (num1, num2) {
  return num1 + num2;
}
```

```
function multiply (num1, num2) {
  return num1 * num2;
}
```

```
function calculator (num1, num2, operator) {  
    return operator (num1, num2);  
}  
calculator (4, 5, add); → 9
```

debugger; → entra en debug mode.

Higher order functions are functions that can take other functions as inputs.

Reproducir audio con JavaScript

```
var audio = new Audio("audio-file.mp3"); } Buscar en MDN DOC.  
audio.play()
```

console.log(this); → tree de elementos en escritorio.

console.log(this.innerHTML);

JavaScript Objects.

```
var bellboy1 = {  
    name: "Thimmy",  
    age: 19,  
    hasWorkPermit: true,  
    languages: ["French", "English"]  
}
```

bellboy1.name → "Thimmy"

Object property

Constructor Functions:

```
function Bellboy (name, age, hasWorkPermit, languages) {  
    this.name = name;  
    this.age = age;  
    this.hasWorkPermit = hasWorkPermit;  
    this.languages = languages;
```

Initialize Object

```
var bellboy1 = new BellBoy("Timmy", 19, true, ["French", "English"]);
```

↑ Capitalized name (not camel case)

```
switch (expression) {  
    case expression:  
        ;  
        break;  
    default:  
}
```

} switch statements.

case-break. encierra el código que se ha de ejecutar.

Si bien default casi nunca se usa es buena práctica agregarle un console.log (expresión), como por ejemplo registrar algún evento inspeccionado.

Métodos de objetos.

```
var bellboy1 = {
```

name: "Timmy",

age: 19,

} propiedades

function anonymous

moveSuitcase: function () {

alert ("May I take your suitcase?");

pickSuitcase();

move();

}

}

} método.

} se incorpora en forma similar al constructor de objetos o func. constructores

bellboy1.moveSuitcase(); → call method.

keypress deprecated → use keydown.

Se agrega el event Listener a todo el documento.

```
document.addEventListener("keydown", function() {
    alert("key was pressed!");
});
```

console.log(event); key → propiedad interesante.

```
function makeSound(key) {
    switch(key) { ... };
}
```

makeSound(button.innerHTML); → para clicks

makeSound(event.key); → para teclado.

Higher Order Functions → funciones que reciben otras funciones como argumentos.

Callback Functions → funciones que sirven como argumentos de las Higher Order Functions.

```
$0. addEventListener("click", function(event) {
    console.log(event);
});
```

Esta función no es llamada por el usuario sino por el objeto que experimenta el click.
Cuando eso pasa le pide una que nos da información sobre el evento.

el evento es pasado como argumento de la función. Este parámetro puede tener cualquier nombre.

```
function anotherAddEventListener(typeOfEvent, callback) {
```

// Detach event code...

```
var eventThatHappened = {
    eventType: "keypress",
    key: "p",
    durationKeyPress: 2
};
```

```
if (eventThatHappened.eventType == typeOfEvent) {
    callback(eventThatHappened);
}
```

Este ayuda a entender el funcionamiento

Callbacks es un concepto difícil de comprender. Poner atención a ello!

```
function buttonAnimation(currentKey) {
    var activeButton = document.querySelector("." + currentKey);
    activeButton.classList.add("xpressed");
    activeButton.classList.remove("pressed");
}
```

← Tiempo de permanencia
↑
setTimeout function

jQuery

document.querySelector("h1") → \$("h1")

minifier.org permite reducir el tamaño de los archivos CSS y JS con el objeto de tener cargar más rápidamente las páginas.

document.querySelector All("button") = \$("button")	→ a jQuery
document.querySelector ("h1") = \$("h1")	
propiedad CSS y JS.	seleccionar uno o todos los elementos.

```
console.log($(".h1").css("font-size")); } mismo resultado
console.log($(".h1").css("fontSize")); }
```

\$(".h1").css("color")	método CSS con un argumento llame al valor de la propiedad (se comporta como getter), con dos argumentos elige valor a la propiedad (se comporta como setter).
\$(".h1").css("color", "red")	

No es buena práctica usar JS para cambiar propiedades de CSS.

Mejor es usar el método addClass.

```
$(".h1").addClass("big-title");
```

para asignar múltiples clases hay que colocarlos dentro de los comillas separados por espacios.

`$("h1").hasClass("clase1");` → devuelve true si el elemento h1 tiene la clase1, caso contrario devuelve false.

`$("button").text("nuevo texto");` cambia el texto del o los elementos seleccionados.

`$("button").html("Hey ");` → equivale al innerHTML.

A utilizar métodos, jquery permite hacer lo mismo que en javascript pero con menos código.

`$("a").attr("href");` → devuelve el atributo

`$("a").attr("href", "www.google.com");` → asigna el atributo.

Tener en cuenta que la clase y el id también son atributos.

`$("h1").click(function() {
 $("h1").css("color", "purple");
});` } event listener.

`$("button").click(function() {
 $("h1").css("color", "purple");
});` } aplica un event Listener a todos los botones sin necesidad de un ciclo for.

`$("input").keypress(function(event) {
 console.log(event.key);
});`

`$(document).keypress(function(event) {
 $("h1").text(event.key);
});`

`$("h1").on("evento", function() {
 //acción a ejecutar
});`

Agregar elementos de una página con jQuery

`$("#hf").before(<button>New</button>);` → *

`$("#hf").after(<button>New</button>);`

`$("#hf").prepend(<button>New</button>);` → **

`$("#hf").append(<button>New</button>);`

`$("#button").remove();` → elimina todos los botones

* `<button>New</button><hf>Hello</hf>`

** `<hf><button>New</button>Hello</hf>`

Animaciones con jQuery.

`$("#hf").hide();`

`$("#hf").show();`

`$("#hf").toggle();`

`$("#hf").fadeOut();`

`$("#hf").fadeIn();`

`$("#hf").slideUp();`

`$("#hf").slideDown();`

`$("#hf").slideToggle();` solamente reglas CSS con valores numéricos.

`$("#hf").animate({css rules});`

Se pueden encadenar métodos para que sucedan progresivamente

`$("#hf").slideUp().slideDown().animate({opacity: 0.5});`

Command Line.

`mkdir <nOMBRE DEL DIRECTORIO>` crea un directorio

`touch <nOMBRE DEL ARCHIVO>` crea un archivo

`open -e code <nOMBRE DEL ARCHIVO>` abre el archivo en VS code.
start en Windows

`rm <nOMBRE DEL ARCHIVO>` elimina el archivo.

`pwd` print working directory muestra el directorio donde me encuentro.

`rm *` elimina todos los archivos en el directorio.

↳ solamente elimina archivos, no directorios.

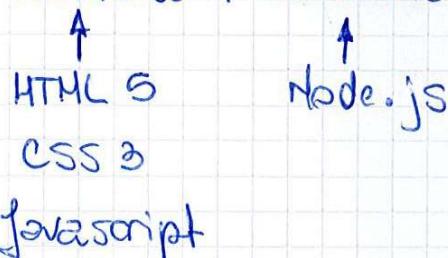
`rm -r <nombre del directorio>` elimina el directorio con todos los archivos y sub-directorios que allí se encuentran.

`rm -rf <nombre del directorio>` elimina el directorio sin pedir confirmación

↳ force

Back end Web Development. con la barra / al final.

Full Stack = Frontend + Backend.



Backend → Server, applications, databases.

Node.js → lenguaje de backend. El framework de este lenguaje es express.

Con Node.js se puede crear una aplicación de escritorio como Atom. Node.js interactúa con el hardware.

En la lección 210 se explica como instalar Node.js en Windows.

`node --version` para confirmar que se haya instalado bien.

`node index.js` ejecuta el archivo index.js en la terminal. de comandos.

REPL Read Evaluation Print Loop.

node ejecutando este comando entra en modo REPL y es como escribir código JS en la consola de Chrome.

En modo REPL, escribiendo parte de un comando y presionando TAB dos veces trae las posibles opciones.

Probar: von + TAB + TAB

console. + TAB + TAB ^C

- exit sale de REPL (ctrl + C dos veces)

Native Node Modules

www.nodejs.org/api/

Por ejemplo, podemos interactuar con el sistema de archivos de la computadora.

const fs = require("fs"); Para usar un módulo hay que requerirlo. En este caso se requiere al módulo File System.

fs. copyFileSync(source, destination); Busca el archivo source por ej.: "file1.txt" ↑ por ej.: "file2.txt"

y lo copia en el archivo destination. Si el archivo destination no existe lo crea, si existe lo reemplaza.

Se pueden usar módulos internos, que se instalan con node.js, o módulos externos, que se descargan.

npm gestor de paquetes de node. Node Package Manager.

npm viene instalado con node.

npm init ejecutado en la terminal inicializa npm

Cuando se inicializa pide una serie de datos. Por defecto volver al minuto 3 de la locación D204.

www.npmjs.com para buscar paquetes. Allí se explica como instalar y usar los módulos.

Hacer una prueba con el módulo superheroes

json formato de archivo que se utiliza para organizar datos.

Probar hacer un programa que imprime en la terminal un supertitrope VS. Supervillano.

Express Framework de Node.js que sirve para organizar el código para aplicaciones web con Node.

Creación de un servidor local utilizando Express

Ir al minuto 3 de la lección 2.0.7 para ver la creación de un servidor con Express. Este es el proceso que hay que repetir cada vez que comience un proyecto de desarrollo web. expressjs.com documentación de express.

En la documentación de Express se encuentran las instrucciones para la instalación.

npm install express instala, aunque la documentación indica
npm install express --save

En la documentación ver el ejemplo Hello world! en el menú Getting started.

const app = express(); el uso de la constante app es una práctica común en el desarrollo web.

Se detiene el servidor express con Ctrl + C.

```
app.listen(3000, function () {  
    console.log("Server started on port 3000.");  
});
```

localhost:3000/ tendría que ser la raíz o página principal de mi sitio web. Algo así como www.ccll.com.ar/ cuando se pone un nombre de página web en el explorador

este envía una solicitud al servidor donde la página se encuentra alojada. Cuando los servidores reciben la solicitud envían una respuesta al cliente. La respuesta son los archivos html, css y js.

home ~~root~~ route

```
app.get("/", function(request, response) {  
    console.log(request); response.send("Hello!");  
});
```

No solamente se puede enviar texto plano, se puede enviar html. response.send("<h1>Hello world!</h1>");

```
app.get("/", function(req, res) {  
    res.send("Hello"); // buenas prácticas en el uso de express.  
});
```

Aquí es donde toman importancia las callbacks y funciones de orden superior.

nodemon paquete npm que facilita auto iniciar los servidores. En caso de tener problemas ir ~~la~~ la dirección. 2169.

```
app.get("/contact", function(req, res) {  
    res.send("Contact me at...");  
});
```

El código anterior responde a la solicitud del explorador expresada como localhost:3000/contact.

Se puede escribir código para muchos routes.

nodemon.io monitorea cambios en el código fuente y reinicia el servidor automáticamente. Una vez que se instala estará disponible para todos los proyectos.

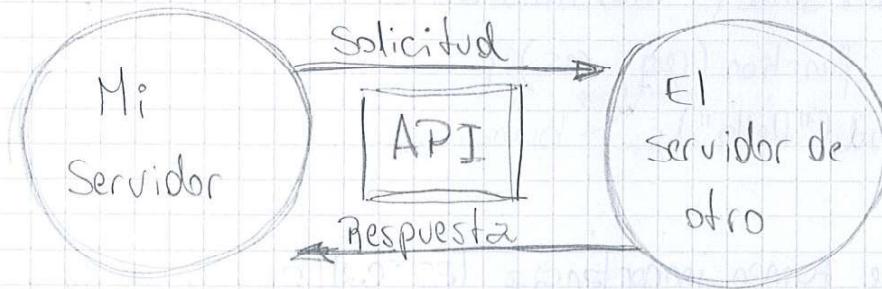
entry point.
nodemon server.js ejecutando en la carpeta del proyecto comienza a monitorizar los cambios.

El entry point es el archivo principal del proyecto.

npm install body-parser permite analizar la información npm i body-parser también que se envía mediante la solicitud función HTTP

API Application Programming Interfaces

Un conjunto de comandos, funciones, protocolos, y objetos que los programadores pueden usar para crear software o interactuar con un sistema externo.



API Endpoints , Paths , Parameters , Authentication.

Endpoint: Todas API que interactúan con un sistema externo como un servidor tiene un "endpoint".

El endpoint de la API kanye.rest es <https://api.kanye.rest>

Path: Cuando se pretende una respuesta más específica de la API.

Parameters: endpoint / path? contains = parameter de pregunta separados por &

Se acepta más de un parámetro separado por &

Probar con la joke API como para entender esto.

API Authentication para limitar el uso de la API

El orden en que se colocan los parámetros de la API no importa.

postman.com logueado con la cuenta de Google es una forma más fácil de checar las APIs. Ver minuto 9 de la lección 24J.

JSON Java Script Object Notation.

Tiene la ventaja que puede ser leído por seres humanos y ser comprimidos para que ocupen poco lugar.

Otros formatos que devuelven las APIs es XML ó HTML.

JSON Viewer Awesome extensión de Chrome que permite ver mejor los formatos JSON.

Datos de autenticación en openweathermap.org.

User:

password:

postman.co logueado con [REDACTED]

Status Code 200 → Ok

formato hexadecimal 404 → página no encontrada.

JSON.parse(data) → convierte datos en formato JSON

JSON.stringify(objet) → aplica información en formato JSON para que ocupe menos espacio.

npm install body-parser express request se instalan varios paquetes ~~en~~ una sola linea.

Desplegar tu una app de Node.js al servidor de Iowebz usando c-panel.

1- Crear un subdominio Me debo asegurar que el nombre del subdominio coincida con el del dominio. Ej.: prueba.ingr.es

2- Set up Node.js App.

3. Agregar los archivos.

4. Add code snippets.

• npmrc crear este archivo en la carpeta raíz.

Archivos → carpeta de la aplicación /home/grobigl3/prueba /
→ carpeta del subdominio /home/public_html/prueba

.htaccess se crea solo. Agrega:

DirectoryIndex disabled

Rewrite Engine On

Rewrite Rule ^\$ http://127.0.0.1:30000/ [?,L]

Rewrite Cond %{REQUEST_FILENAME} !-f

Rewrite Cond %{REQUEST_FILENAME} !-d

Rewrite Rule ^(,*)\$ http://127.0.0.1:30000/\$1 [?,L]

Header set Access-Control-Allow-Origin "*"

5. Force https solo si tengo certificado SSL

6. Run the scripts.

Mailchimp

Usuario:

[REDACTED]

Contraseña:

Cuando tengo el archivo "package.json" que contiene las dependencias solamente debo ejecutar "npm install" para que se instalen todos los módulos de Node.

CDN = Content Delivery Network.

[REDACTED]

e-mail

Flexbox

Opciones para el posicionamiento de elementos en páginas web.