

Introducción a la web: JavaScript

II C1005 - Computación: Ciencia y Tecnología del Mundo Digital

Fernando Florenzano Hernández
faflorenzano@ing.puc.cl

<https://github.com/fdoflorenzano/intro-a-la-web-2>

Contenidos

Motivación

Historia

JavaScript

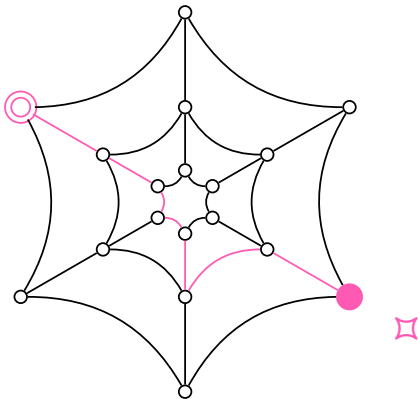
Contenidos

Motivación

Historia

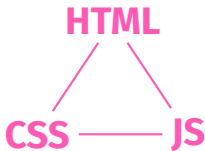
JavaScript

Recordando la Web



Desarrollo web

El **desarrollo web** define la creación de **sitios web**. Para conseguirlo se hace uso de tecnologías de **software** del lado del **servidor** y del **cliente**. Del lado del cliente, hay tres tecnologías base:



Hoy aprenderemos sobre **JavaScript**.

Motivación

Ya aprendimos sobre **HTML** y **CSS**, ¿qué le falta a nuestras **páginas web**?

¡Interacción y dinamismo!

Otro **estándar de la web** hoy en día, es el agregar interacción y dinamismo a sus páginas. Le agrega **valor** de interfaz para el **mejor uso** de parte de su usuario.

Contenidos

Motivación

Historia

JavaScript

Historia de JavaScript

- **1995**: JavaScript fue diseñado en sólo **diez días**, por Brendan Eich, cuando trabajaba en **Netscape**.
- Inicialmente, se llamó **Mocha**, luego **LiveScript** y finalmente, por asuntos de marketing... **JavaScript**.
- Entre **1996-1999**: Se hace el intento de estandarizar JavaScript por ECMA International con el propósito de que todos los navegadores implementaran el mismo lenguaje. Comienza la **browser wars**.
- Este estándar se llama **EcmaScript** (abreviado **ES**).

Historia de JavaScript

- **2010-2015:** Surgen los *frameworks* de JavaScript: **Backbone**, **Ember**, **AngularJS**, **React**, **Angular** y **Vue**.



- En el mismo tiempo, aparecen *frameworks* y librerías para desarrollo móvil y de escritorio basado en JavaScript como: **ionic**, **React Native** y **Electron**.



Historia de JavaScript

- En **2009** se libera **ES5**, versión con muchas mejoras que todos los navegadores soportan.
- En **2015** se libera **ES6** que introduce **muchas** nuevas funcionalidades. Esta es la versión que aprenderemos hoy.
- Desde entonces, cada año ha salido una especificación nueva con algunas novedades: **ES7 (2016)**, **ES8 (2017)** y **ES9 (2018)**.

Contenidos

Motivación

Historia

JavaScript

JavaScript

- Es un lenguaje de programación de **alto nivel**, **interpretado** y **multi-paradigma**, como **Python**.
- Es el **único** lenguaje que se puede ejecutar en un **navegador web**.
- Es un lenguaje por si solo, pero que también permite interactuar con la **estructura** de una página web y **alterarla**.

JavaScript

Variables

```
1  const numero = 1005; // una constante
2  const curso = 'Computación: Ciencia y ...';
3  let hora = 10; // variable que cambia
4  hora = 11;
```

```
1  const numero = 1005; // una constante
2  numero = 1006;
3  TypeError: Assignment to constant variable.
```

```
1  var numero = 1005; // funciona, pero no nos gusta var
```

JavaScript

Control de flujo

```
1  const cierto = true;
2  const falso = false;
3
4  if (cierto && falso) {
5      console.log('Esto no se imprimirá.');
```

6 } else if (cierto || falso) {

```
7      console.log('Esto sí se imprimirá.');
```

8 } else {

```
9      console.log('Esto tampoco se imprimirá.');
```

10 }

JavaScript

strings

```
1  const cadena1 = 'Computación: ';
2  const cadena2 = 'Ciencia y Tecnología ';
3  const cadena3 = 'del Mundo Digital';
4  const cadenaCompleta = cadena1 + cadena2 + cadena3;
5  const largo = cadenaCompleta.length; // 51
```

JavaScript

template strings

```
1  const texto = 'IIC1005 tiene largo: ' + largo;  
2  console.log(texto);
```

```
1  // ES6  
2  const texto = `IIC1005 tiene largo: ${largo}`;  
3  console.log(texto);
```


JavaScript

arrays

```
1  let listaDePi = [3, 1, 4, 1, 5];
2  listaDePi.push(9); // [3, 1, 4, 1, 5, 9]
3  listaDePi.pop(); // [3, 1, 4, 1, 5]
4  const largo = listaDePi.length; // 5

1  const primerElemento = listaDePi[0]; // 3
2  const ultimoElemento = listaDePi[listaDePi.length - 1];
3  // 5
4  const segundoYTercero = listaDePi.slice(1, 3);
5  // [1, 4]
6  const terceroEnAdelante = listaDePi.slice(2);
7  // [4, 1, 5]
8  const ultimosDos = listaDePi.slice(-2); // [1, 5]
```

JavaScript

for loops

```
1  for (let index = 0; index <= listaDePi.length; index++) {  
2      console.log(listaDePi[index]);  
3  }
```

```
1  // ES6  
2  for (const numero of listaDePi) {  
3      console.log(numero);  
4  }  
5  for (const caracter of 'Computación: Ciencia...') {  
6      console.log(caracter);  
7  }
```

JavaScript

objects

```
1  const profesor = {
2      nombre: 'Denis',
3      apellido: 'Parra',
4      cursos: 3
5  };
6  const nombreProfe = profesor['nombre']; // 'Denis'
7  const cursosProfe = profesor.cursos; // 3

1  // ES6
2  for (const propiedad in profesor) {
3      console.log(`${propiedad}: ${profesor[propiedad]}`);
4  }
5  // nombre: Denis
6  // apellido: Parra
7  // cursos: 3
```

JavaScript

destructuring (ES6)

```
1  const tecs = ['HTML', 'CSS', 'JS'];
2  const [tec1, tec2, tec3] = tecs;
3  console.log(tec3); // 'JS'
4  const [primer, ...resto] = tecs;
5  console.log(resto); // ['CSS', 'JS']
```

```
1  const profeDeHoy = {
2      nombre: 'Fernando',
3      apellido: 'Florenzano',
4      cursos: 1
5  };
6  const {nombre, apellido} = profeDeHoy;
7  console.log(nombre); // 'Fernando'
```

JavaScript

functions

```
1  function sumaCinco(numero) {  
2      const resultado = numero + 5;  
3      return resultado;  
4  }  
5  sumaCinco(3); // 8
```

```
1  const suma = function (numero1, numero2) {  
2      const resultado = numero1 + numero2;  
3      return resultado;  
4  }  
5  suma(3, 5); // 8
```

JavaScript

arrow functions (ES6)

```
1  const sumaCinco = (numero) => {  
2      const resultado = numero + 5;  
3      return resultado;  
4  }  
5  sumaCinco(3); // 8
```

```
1  const suma = (numero1, numero2) => numero1 + numero2;  
2  suma(3, 5); // 8
```

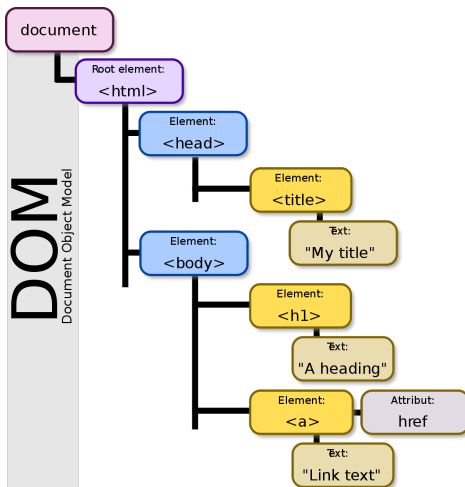
DOM

Document Object Model

El **DOM** es una interfaz que, a través de una representación estructurada, permite acceder y manipular un documento **HTML**.

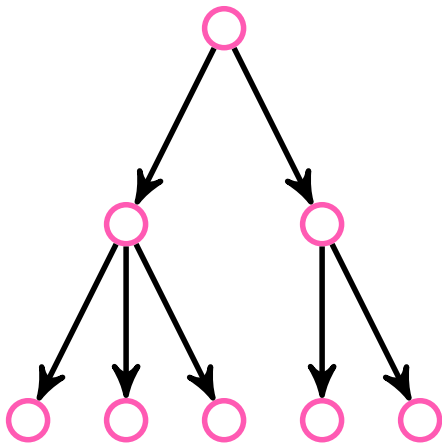
Esta representación se modela como un **árbol**, en donde cada **nodo** es un **objeto** del documento.

DOM

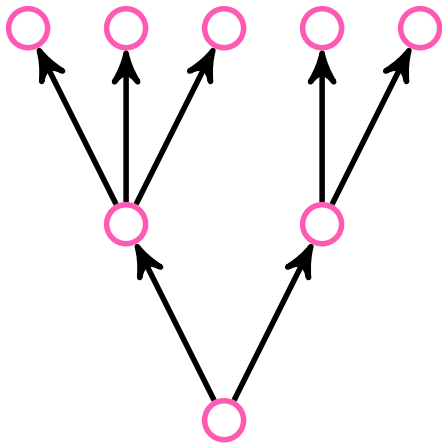


Fuente: **Wikipedia: DOM**

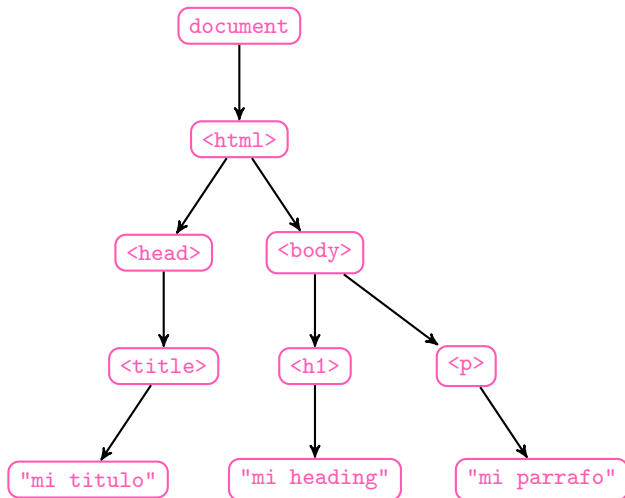
Árboles y el DOM



Árboles y el DOM



Árboles y el DOM



Importar JS en HTML

De forma similar a CSS, hay etiquetas para escribir directamente o para importar código **JavaScript** al documento HTML:

```
<script>  
const elProyectorEs = 'malo';  
const vecesQueHaPestornado = 1304;  
</script>
```

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

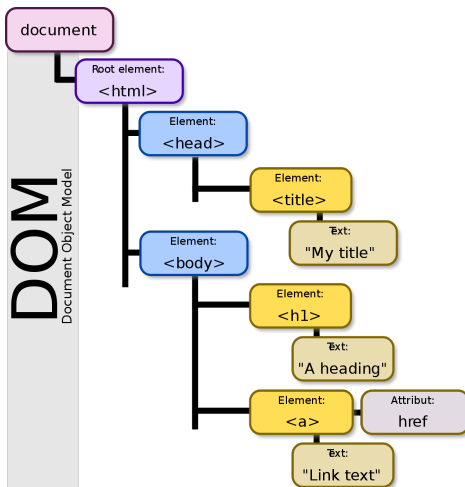
JavaScript y el DOM

Podemos acceder al **DOM** a través del objeto **document**.

```
1 console.log(document.URL); // imprime el URL actual
```

```
1 document.children; // contiene los elementos hijos  
2 // en este caso: [<html>]
```

JavaScript y el DOM



Fuente: **Wikipedia: DOM**

JavaScript: elementos

```
1  const elemento = document.getElementById('idElemento');  
2  // retorna elemento en el documento de id='idElemento'
```

```
1  const nuevoP = document.createElement('p');  
2  // retorna nuevo elemento de etiqueta <p>
```

```
1  elemento.appendChild(nuevoP);  
2  // ahora, <p> es hijo del elemento inicial
```

```
1  elemento.removeChild(nuevoP);  
2  // <p> fue eliminado como hijo del elemento  
3  // y del documento entero :(
```

JavaScript: elementos de texto

```
1  const nuevoP = document.createElement('p');
2  const nodoDeTexto = document.createTextNode('¿Qué tal?');
3  nuevoP.appendChild(nodoDeTexto);
4  // Eso entregaría una elemento de la forma:
5  // <p>¿Qué tal?</p>
```


JavaScript: clases

```
1 elemento.className = 'magenta';  
2 // ahora el elemento tiene clase: 'magenta'  
3 // en realidad, tiene clases: ['magenta']
```

```
1 elemento.classList.add('azul');  
2 // ahora el elemento tiene clases: ['magenta', 'azul']
```

```
1 elemento.classList.remove('magenta');  
2 // ahora el elemento tiene clases: ['azul']
```

JavaScript: eventos

```
1 elemento.addEventListener('click', () => {  
2     // código que se ejecuta cada vez que  
3     // se hace clic sobre elemento  
4 });
```

JavaScript: ejemplo

```
1  const contenedor = document.getElementById('contenedor');
2  const boton = document.getElementById('boton');
3  boton.addEventListener('click', () => {
4      const nuevoH1 = document.createElement('h1');
5      const textoH1 = document.createTextNode(';Hola!');
6      nuevoH1.appendChild(textoH1);
7      contenedor.appendChild(nuevoH1);
8  });
9  // ¿Qué ocurriría aquí?
```

Aprender practicando

Para el material:

- **Descarga** el repositorio

<https://github.com/fdoflorenzano/intro-a-la-web-2>.

- En la carpeta **ejemplo** encontrarán el ejemplo de la clase de hoy.
- En la carpeta **practico** encontrarán un ejercicio para ustedes.
- **Edita** los archivos y revisa los cambios que se producen.

Para trabajar:

- Usa un navegador web **moderno**, como **Firefox** o **Chrome**.
- Descarga un editor de texto con muchas funcionalidades. Opciones populares son **Atom**, **Visual Studio Code**, **Notepad ++** y **Sublime Text**.

Herramientas útiles y referencias de JS

- Los **developer tools** del navegador, en este caso, revisar la **consola** por cualquier `console.log()`.
- **Can I Use... ?**
- **Tutoriales web** del **MDN**.
- **Tutoriales web de JavaScript** del **MDN**.