
PUCP

Desarrollo Web con Python

CETAM - PUCP

Sesión 7:

- Javascript
  - Definición de variables
  - Operaciones aritméticas
  - Eventos en el DOM
  - Integración de javascript con HTML y CSS
  - GET y POST

Javascript

Desarrollo Web con Python - Sesión 7

PUCP

Javascript

JavaScript es un lenguaje de programación ampliamente utilizado en el desarrollo web para crear interactividad y dinamismo en las páginas. Es un lenguaje de alto nivel que se ejecuta en el navegador del usuario, lo que permite manipular elementos HTML, realizar solicitudes a servidores, responder a acciones del usuario y actualizar el contenido de manera dinámica. Al ser interpretado por el navegador, JavaScript permite construir experiencias interactivas en tiempo real, como formularios interactivos, animaciones y actualizaciones de contenido sin necesidad de recargar la página. Su versatilidad y compatibilidad con múltiples plataformas lo convierten en una herramienta esencial para el desarrollo web moderno.

Funcion console.log y alert

Las funciones console.log() y alert() permiten mostrar información de javascript en diferentes ubicaciones:

```
console.log('hola')
alert('Ingrese otro tipo de valor')
```

Definición de variables

La declaración de variables se da utilizando las palabras reservadas 'var' y 'let':

```
var x = 5
let y = 6
let z = x + y
```

'var' es un identificador de variables antiguo mientras que 'let' es más actual, se recomienda el uso de 'let' ya que permite un mejor flujo del programa.

Tipos de variables

Los tipos de variables principales en javascript son:

- Numéricas
- Cadenas de texto
- Arreglos
- Booleanos

La verificación del tipo de una variable se puede hacer con typeof()

Conversión de variables

Para convertir variables de diferentes tipos se utilizan las funciones:

- Number() -> Convertir cadena de texto a variable numérica
- String() -> Convertir variable numérica a cadena de texto

Definición de objetos en Javascript

Los objetos en javascript se definen con la siguiente estructura:

```
let person = {
    nombre: 'alexander'
    apellido: 'segovia'
    edad: '25'
}
```

Operaciones aritméticos

Los operadores aritméticos en javascript son:

- Adición (+)
- Sustracción (-)
- Multiplicación (*)
- División (/)
- Exponenciación (**)
- Módulo (%)
- Incremento (++)
- Decremento (--)

Operadores lógicos

Los operadores lógicos en javascript son:

- Igualdad en valor (==)
- Igualdad en tipo y valor (===)
- Desigualdad (!=)
- Desigualdad en tipo y valor (!==)
- Mayor que (>)
- Menor que (<)
- Mayor o igual que (>=)
- Menor o igual que (<=)

Sentencias selectivas: if

```
let a = 25
let b = 43
let c = '43'
if(a === b)
{
    console.log('a y b son iguales ')
}
if( b == c)
{
    console.log('b y c son iguales?')
}
```

Sentencias iterativas: for

```
usuarios_sistema = ['javier','alexander','martin','diego']

for(let i = 0; i < usuarios_sistema.lenght; i++)
{
    console.log(usuarios_sistema[i])
}
```

Sentencias selectivas: while

```
let i = 35
let counter = 0
while(counter < i)
{
    counter = counter + 1
}
```

Eventos en el DOM

Los diferentes eventos que pueden ser mapeados en el DOM son:

- onclick
- onchange
- onmouseover
- onkeyup
- onkeydown
- ...

Ejemplos de aplicación

Aplicando lo conceptos revisados se procede a ejecutar el siguiente ejemplo:

Implementar un contador presentando la información en diferentes formas (consola, alert …)

Integración con elementos HTML y CSS

Los diferentes eventos que pueden ser mapeados en el DOM son:

- document.querySelector('.class')
- document.querySelector('tag')
- document.querySelector('#id')

Integración con elementos HTML y CSS

Otras formas de seleccionar elementos muy utilizadas son:

- document.getElementById('id')
- document.getElementsByName()
- document.querySelectorAll('tag')

Ejemplos de aplicación

Con javascript se desarrolla un programa que permite agregar números editables a una lista para finalmente sumarlos y presentarlo en un mensaje al final del HTML

¿Qué sigue?

Front End

HTML

CSS

JS

Bootstrap

Back End

Flask
web development,
one drop at a time

django

Express JS

Laravel

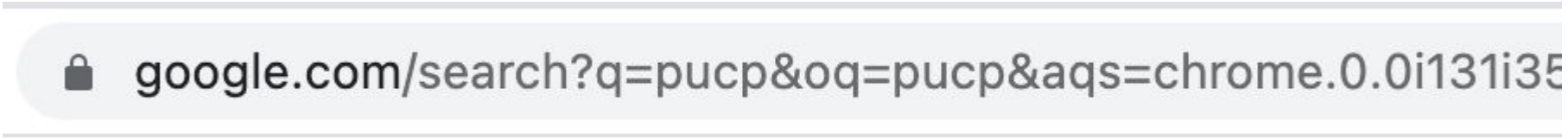
¿Cómo integramos lo aprendido?

El formato JSON

```
{
    "rates": {
        "EUR": 0.907,
        "JPY": 109.716,
        "GBP": 0.766,
        "AUD": 1.479
    },
    "base": "USD"
}
```

Retornar textos en JSON

El formato JSON nos permite transferir información entre el servidor backend y las vistas entregadas en el front end. Es posible desplegar la totalidad de la información en un documento HTML; sin embargo esto no es eficiente y sobrecarga el navegador. Una buena práctica es solo renderizar lo importante y transferir la información en formato JSON y modificar el DOM en el lado del usuario.



Peticiones AJAX (Get)

Las peticiones AJAX nos permiten extraer información del servidor sin la necesidad de cambiar de vista. Esto se logra gracias al paso de información en formato JSON.

```
fetch(` /post?start=${start}&end=${end}` )
.then(response => rensponse.json())
.then(data => {
    console.log(data)
})
```

Peticiones AJAX (Post)

También se tienen peticiones ajax que nos permiten enviar información hacia el servidor y almacenarla en base de datos ejecutando una acción con el método POST:

```
fetch(url,
{
	method:"POST",
	headers:
	{
		"X-Requested-With": "XMLHttpRequest",
		"X-CSRFToken": getCookie("csrftoken"),
	},
	body:JSON.stringify(ejemplo)
})
.then(response => response.json())
.then(data => {
	console.log(data)
})
```

Peticiones AJAX (Post)

Retorno del token csrf al momento de utilizar el posteo de información utilizando ajax.

```
function getCookie(name)
{
	let cookieValue = null;
	if (document.cookie && document.cookie !== "")
	{
		const cookies = document.cookie.split(";");
		for (let i = 0; i < cookies.length; i++)
		{
			const cookie = cookies[i].trim();
			if (cookie.substring(0, name.length + 1) === (name + "="))
			{
				cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
				break;
			}
		}
	}
	return cookieValue;
}
```

Gracias