Variables y operadores

Curso de Fundamentos

Introducción

En las secciones anteriores aprendiste a estructurar páginas web con HTML y a darles estilo con CSS. El siguiente paso es hacer que la página web sea *interactiva* , que es exactamente para lo que sirve JavaScript.

En esta sección, nos centraremos en los fundamentos de JavaScript y cómo usarlo para manipular todas las diversas interacciones entre la página web y el usuario.

Resumen de la lección

Esta sección contiene una descripción general de los temas que aprenderá en esta lección.

- Ejecutar código JavaScript utilizando un archivo HTML.
- Declarar variables con `let` y `const` .
- Realizar operaciones numéricas.
- Realizar operaciones con cadenas.
- Utilizando operadores lógicos y matemáticos.

Cómo ejecutar código JavaScript

Todo el código JavaScript que escribiremos en la mayor parte del curso Fundamentos se ejecutará a través del navegador. Las lecciones posteriores de Fundamentos y la

ruta NodeJS le mostrarán cómo ejecutar JavaScript fuera del entorno del navegador.

Fuera de estas lecciones, por ahora siempre debes ejecutar JavaScript en el navegador de forma predeterminada a menos que se especifique lo contrario; de lo contrario, puedes encontrarte con errores inesperados.

La forma más sencilla de empezar es crear un archivo HTML con el código JavaScript dentro. Utilice el fragmento de código de VS Code `!` + `TAB` para crear el esqueleto HTML básico en un archivo en algún lugar de su computadora. Asegúrese de incluir la `<script>` etiqueta:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-s
  <title>Document</title>
</head>
<body>

  <script>
    // Your JavaScript goes here!
    console.log("Hello, World!")
  </script>

</body>
</html>
```

Guarde y abra este archivo en un navegador web y luego abra la consola del navegador:

1. Haga clic derecho en la página web en blanco.
2. Haga clic en “Inspeccionar” o “Inspeccionar elemento” para abrir las Herramientas para desarrolladores.
3. Busque y seleccione la pestaña “Consola”, donde debería ver el resultado de nuestra `console.log` declaración.

> Vista previa en vivo
>
> Puede utilizar [la extensión Live Preview en Visual Studio Code] para actualizar automáticamente el navegador cuando guarde su archivo en lugar de tener que actualizar manualmente la página para ver los cambios cuando edite su código. ¡Intente editar el texto para decir algo diferente!

`console.log()` es el comando para imprimir algo en la consola de desarrollador de su navegador. Puede usarlo para imprimir los resultados de cualquiera de los siguientes artículos y ejercicios en la consola. Lo alentamos a que codifique junto con todos los ejemplos de esta y futuras lecciones.

Otra forma de incluir JavaScript en una página web es mediante un script externo. Esto es muy similar a vincular documentos CSS externos a su sitio web.

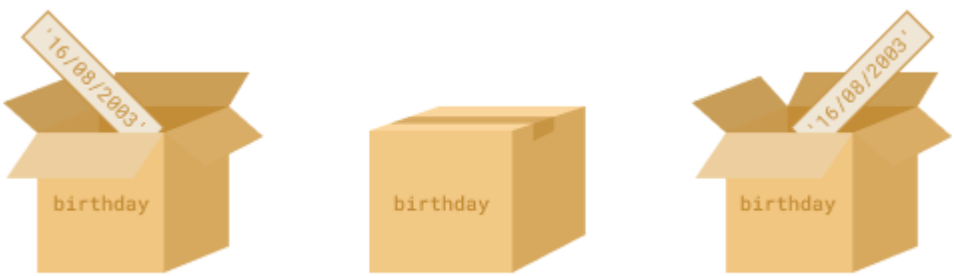
```
<script src="javascript.js"></script>
```

Los archivos JavaScript tienen una extensión `.js` similar a `.css` la de las hojas de estilo. Los archivos JavaScript externos se utilizan para scripts más complejos.

Hemos puesto un nombre a nuestro archivo `javascript.js` , pero podríamos haber elegido cualquier nombre, como por ejemplo `my-script.js` , o incluso ninguno `.js` . Lo que realmente importa es la `.js` extensión.

Variables

These are the building blocks of any program, you can think of variables as “storage containers” for data in your code.



You can declare variables using the `let` keyword. Let's try it! (No pun intended).

```
let name = "John";
let surname = "Doe";

console.log(name);
console.log(surname);
```

What will the `console.log` output? Try it out!

You can also re-assign variables:

```
let age = 11;
console.log(age); // outputs 11 to the console

age = 54;

console.log(age); // what will be output now?
```

Notice the lack of `let` on line 4 - we don't need it since the variable has already been *declared* earlier and we are just re-assigning it here!

Re-assigning is cool and all, but what if we *don't* want it to happen? For example we might have a *constant* `pi` which will never need to be re-assigned. We can accomplish this using the `const` keyword.

```
const pi = 3.14;
pi = 10;

console.log(pi); // What will be output?
```

Your intuition may tell you that `3.14` will be output. Try it!

An error is thrown. It doesn't even reach the `console.log`! You may wonder why we would *want* an error in our code. Truth be told, errors are incredibly helpful at telling us

what is wrong with our code and exactly where the issue is. Without them, our code would still not do what we may want it to, but it would be a major pain to try and find what's wrong!

So in summary, there are two ways to declare a variable:

- `let`, which we can re-assign.
- `const` which we **can't** re-assign and will throw an error if we try.

There is also a third way, `var`, which was the original way variables were declared in JavaScript. `var` is similar to `let` in that variables assigned this way can be reassigned, but it has other quirks that were cleared up when the language introduced `let` and `const`. By and large, it is not used anymore. However, you will likely come across code which uses `var` at some point, so it is useful to know that it exists.

Numbers

Numbers are the building blocks of programming logic! In fact, it's hard to think of any useful programming task that doesn't involve at least a little basic math... so knowing how numbers work is obviously quite important. Luckily, it's also fairly straightforward.

If you went to school, you will likely not find the concept too difficult to grasp. For example, the mathematical expression `(3 + 2) - 76 * (1 + 1)` is also valid JavaScript. If you put that into a `console.log`, it'll evaluate the expression and output the correct number. Try it!

Assignment

Try the following exercises by adding code to a script tag in your HTML file:

1. Add 2 numbers together! In your script, type in `console.log(23 + 97)`. Running this should log `120`.
2. Do the same thing but add 6 different numbers together.
3. Now log the value of the following expression: `(4 + 6 + 9) / 77`. The console should log approximately `0.24675`.
4. Let's use some variables!
    1. Add this statement to the script tag: `let a = 10`.

2. Below it, add `console.log(a)`. When you run this, the browser console should log `10`.
3. Afterwards, re-assign `a` with a different number value. Log `a` again afterwards and it should show the updated value (the previous log should still show the old value of `10` since that was before `a` was re-assigned).
4. Now add to the bottom of the script `let b = 7 * a`.
5. Log what `b` is. It should log the result of 7 multiplied by whatever you re-assigned `a` with.

5. Try this sequence:
   1. Declare a const variable `max` with the value `57`.
   2. Declare another const variable `actual` and assign it `max - 13`.
   3. Declare another const variable `percentage` and assign it `actual / max`.
   4. Now if you log `percentage`, you should see a value in the console like `0.7719`.
6. Take a few minutes to keep playing around with various things in your script tag. Eventually, we will learn how to actually make these things show up on the webpage, but all of this logic will remain the same. Make sure you're comfortable with it before moving on.

Go through the following articles to deepen your knowledge.

1. Read up on variables in JavaScript from JavaScript.info.
2. This W3Schools lesson on JavaScript arithmetic followed by this on JavaScript numbers, are good introductions to what you can accomplish with numbers in JavaScript.
3. This MDN article on JavaScript math covers the same info from a slightly different point of view, while also teaching you how to apply some basic math in JavaScript. There's much more that you can do with numbers, but this is all you need at the moment.
4. Read through (and code along with!) this article on JavaScript operators. Don't forget to do the "Tasks" at the bottom of the page! It will give you a

pretty good idea of what you can accomplish with numbers (among other things!) in JavaScript.

Knowledge check

The following questions are an opportunity to reflect on key topics in this lesson. If you can't answer a question, click on it to review the material, but keep in mind you are not expected to memorize or master this knowledge.

- What three keywords can you use to declare new variables?
- Which of the three variable declarations should you avoid and why?
- What rules should you follow when naming variables?
- What happens when you add numbers and strings together?
- How does the Modulo (%), or Remainder, operator work?
- What's the difference between `==` and `===`?
- When would you receive a `NaN` result?
- How do you increment and decrement a number?
- What's the difference between prefixing and postfixing increment/decrement operators?
- What is operator precedence and how is it handled in JS?
- How do you access developer tools and the console?
- How do you log information to the console?
- What does unary plus operator do to string representations of integers? eg. +"10"

Additional resources

Esta sección contiene enlaces útiles a contenido relacionado. No es obligatoria, por lo que se la puede considerar complementaria.

- El artículo "¿Qué es JavaScript?" de MDN explica un poco más sobre el tema a un alto nivel.

Mejorar en GitHub Informar un problema Ver registro de cambios de la lección

Ver curso

Marcar como completado

Próxima lección

¡Apóyanos!

El Proyecto Odin está financiado por la comunidad. ¡Únase a nosotros para ayudar a estudiantes de todo el mundo apoyando el Proyecto Odin!

Más información Dona ahora

THE ODIN PROJECT

Educación en codificación de alta calidad mantenida por una comunidad de código abierto.

Sobre nosotros

Acerca de

Equipo

Guías

Guías de la comunidad

Guías de instalación

Blog

Casos de éxito

Apoyo

Preguntas frecuentes

Contribuir

Contáctenos

Legal

Términos

Privacidad