기초 웹

22 Winter CNU 기초 스터디

21 남정연
21 박준서

Programming vs Marking Up

What is JS?

JavaScript 자바스크립트

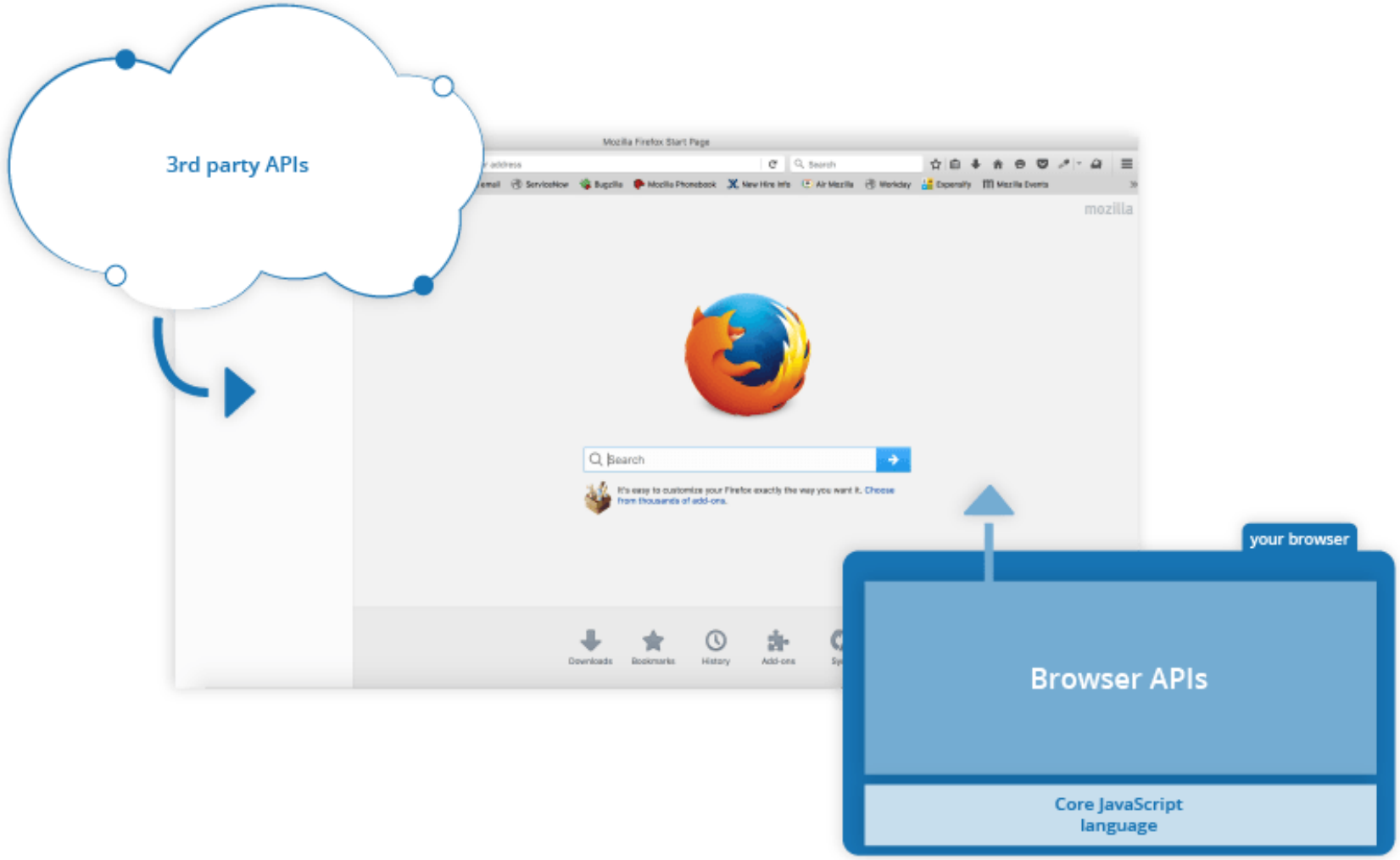
A scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.

What is API?

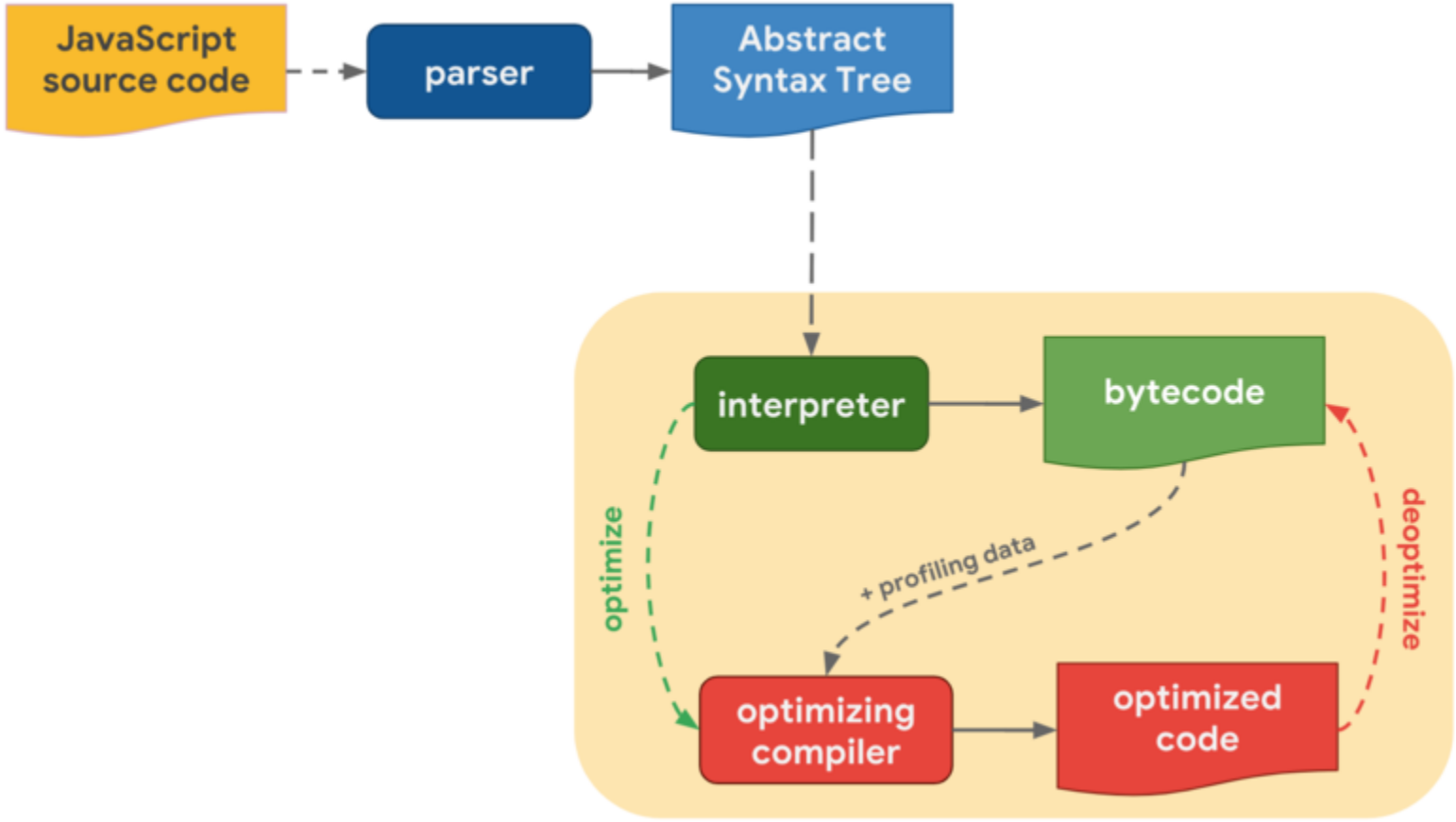
Application Programming Interfaces

응용 프로그래밍 인터페이스

What is API?

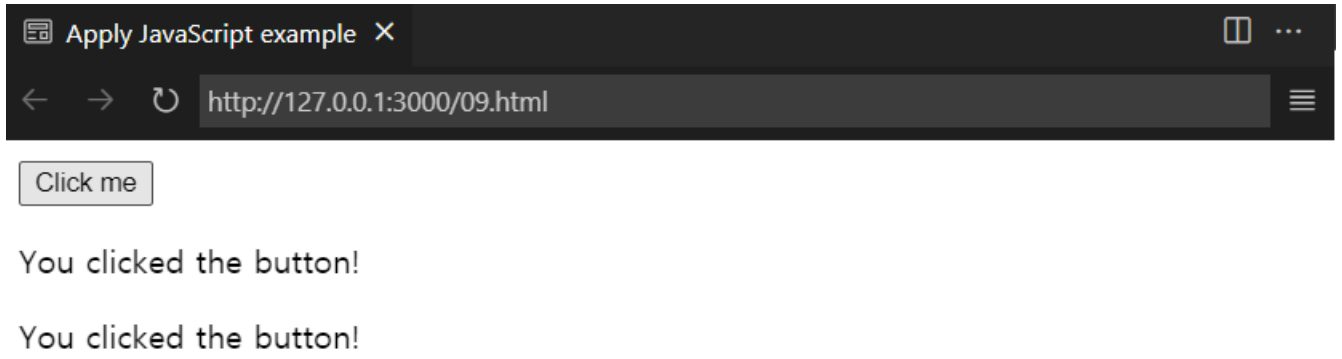

3rd party APIs
Mozilla Firefox Start Page
mozilla
Search
Downloads
Bookmarks
History
Add-ons
your browser
Browser APIs
Core JavaScript
language

Just-In-Time Compile



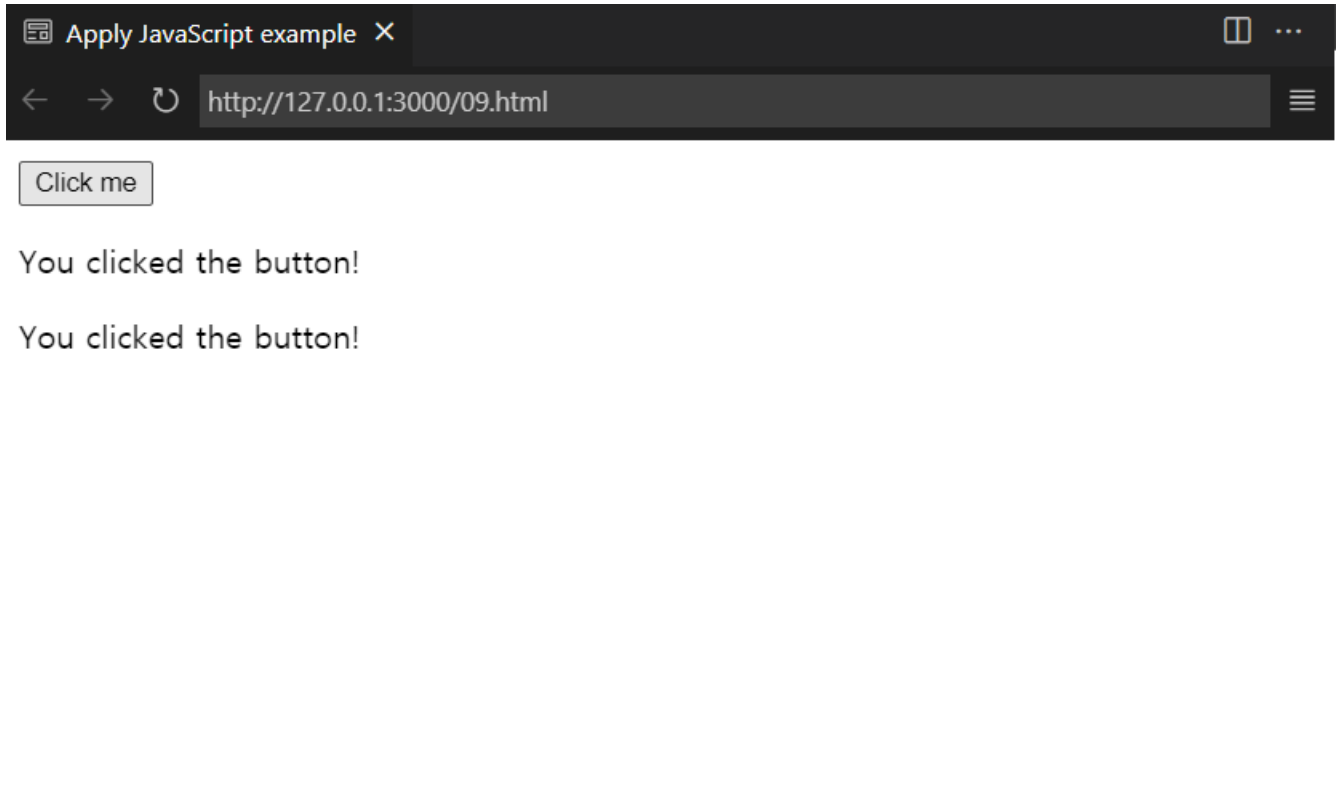
Internal JavaScript

```html
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <title>Apply JavaScript example</title>

    <script>

      document.addEventListener('DOMContentLoaded', () => {
        function createParagraph() {
          const para = document.createElement('p');
          para.textContent = 'You clicked the button!';
          document.body.appendChild(para);
        }

        const buttons = document.querySelectorAll('button');

        for (const button of buttons) {
          button.addEventListener('click', createParagraph);
        }
      });

    </script>

  </head>
  <body>
    <button>Click me</button>
  </body>
</html>
```



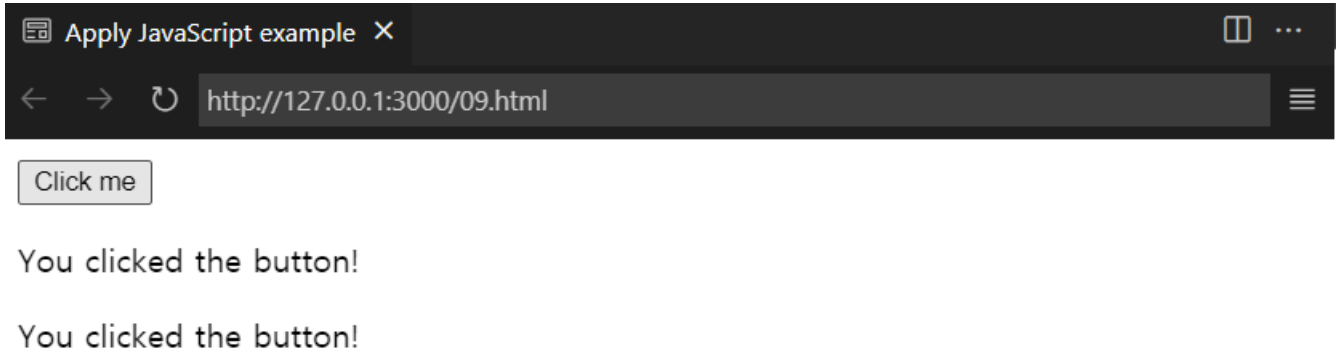
External JavaScript

```html
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <title>Apply JavaScript example</title>

    <script src="script.js" defer></script>

  </head>
  <body>
    <button>Click me</button>
  </body>
</html>
```



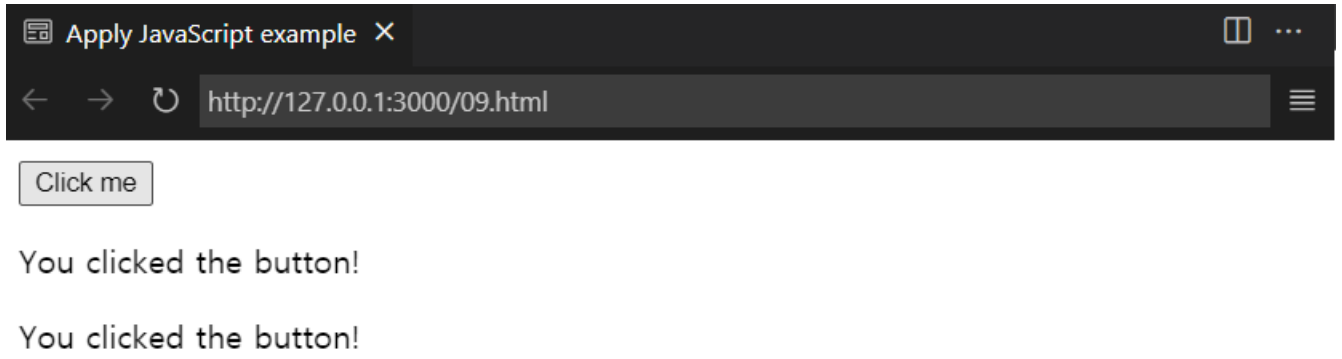
Inline JavaScript Handlers

```html
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <title>Apply JavaScript example</title>

    <script>

      function createParagraph() {
        const para = document.createElement('p');
        para.textContent = 'You clicked the button!';
        document.body.appendChild(para);
      }

    </script>

  </head>
  <body>
    <button onclick="createParagraph()">Click me!</button>
  </body>
</html>
```



addEventListener

```html
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8">
    <title>Apply JavaScript example</title>

    <script>

      document.addEventListener('DOMContentLoaded', () => {
        function createParagraph() {
          const para = document.createElement('p');
          para.textContent = 'You clicked the button!';
          document.body.appendChild(para);
        }

        const buttons = document.querySelectorAll('button');

        for (const button of buttons) {
          button.addEventListener('click', createParagraph);
        }
      });

    </script>

  </head>
  <body>
    <button>Click me</button>
  </body>
</html>
```



Script loading

Script loading

- `async` and `defer` both instruct the browser to download the script(s) in a separate thread, while the rest of the page (the DOM, etc.) is downloading, so the page loading is not blocked during the fetch process.
- scripts with an `async` attribute will execute as soon as the download is complete. This blocks the page and does not guarantee any specific execution order.
- scripts with a `defer` attribute will load in the order they are in and will only execute once everything has finished loading.
- If your scripts should be run immediately and they don't have any dependencies, then use `async`.
- If your scripts need to wait for parsing and depend on other scripts and/or the DOM being in place, load them using `defer` and put their corresponding `<script>` elements in the order you want the browser to execute them.

Comments

- A single line comment is written after a double forward slash (//), e.g.

```
// I am a comment
```

- A multi-line comment is written between the strings /* and */, e.g.

```
/*
  I am also
  a comment
*/
```

Variables

```
var myName = 'Chris';
var myName = 'Bob';
```

```
let myName = 'Chris';
let myName = 'Bob';
```

Variables

```
var myName = 'Chris';
var myName = 'Bob';
```

```
let myName = 'Chris';
let myName = 'Bob';
```

Var hoisting

```
bla = 2;
var bla;

// ...is implicitly understood as:

var bla;
bla = 2;
```

Var hoisting

```
function do_something() {
  console.log(bar); // undefined
  var bar = 111;
  console.log(bar); // 111
}

// ...is implicitly understood as:

function do_something() {
  var bar;
  console.log(bar); // undefined
  bar = 111;
  console.log(bar); // 111
}
```

Variable Types

Numbers

You can store numbers in variables, either whole numbers like 30 (also called integers) or decimal numbers like 2.456 (also called floats or floating point numbers). You don't need to declare variable types in JavaScript, unlike some other programming languages. When you give a variable a number value, you don't include quotes:

```
let myAge = 17;
```

Variable Types

Strings

Strings are pieces of text. When you give a variable a string value, you need to wrap it in single or double quote marks; otherwise, JavaScript tries to interpret it as another variable name.

```
let dolphinGoodbye = 'So long and thanks for all the fish';
```

Variable Types

Booleans

Booleans are true/false values — they can have two values, `true` or `false`. These are generally used to test a condition, after which code is run as appropriate. So for example, a simple case would be:

```
let iAmAlive = true;
```

Whereas in reality it would be used more like this:

```
let test = 6 < 3;
```

Variable Types

Arrays

An array is a single object that contains multiple values enclosed in square brackets and separated by commas. Try entering the following lines into your console:

```
let myNameArray = ['Chris', 'Bob', 'Jim'];
let myNumberArray = [10, 15, 40];
```

Once these arrays are defined, you can access each value by their location within the array. Try these lines:

```
myNameArray[0]; // should return 'Chris'
myNumberArray[2]; // should return 40
```

Variable Types

Objects

In programming, an object is a structure of code that models a real-life object. You can have a simple object that represents a box and contains information about its width, length, and height, or you could have an object that represents a person, and contains data about their name, height, weight, what language they speak, how to say hello to them, and more.

Try entering the following line into your console:

```
let dog = { name : 'Spot', breed : 'Dalmatian' };
```

To retrieve the information stored in the object, you can use the following syntax:

```
dog.name
```

Dynamic typing

```
let myNumber = '500'; // oops, this is still a string
typeof myNumber;
myNumber = 500; // much better — now this is a number
typeof myNumber;
```

Constants

```
let count = 1;
count = 2;
```

```
const count = 1;
count = 2;
```

Functions

```
function checkGuess() {
  alert('I am a placeholder');
}
```

```
checkGuess();
```

Operators

Operator	Name	Example
+	Addition	`6 + 9`
-	Subtraction	`20 - 15`
*	Multiplication	`3 * 7`
/	Division	`10 / 5`

Operators

Operator	Name	Example

`===`	Strict equality (is it exactly the same?)	`5 === 2 + 4 // false`
		`'Chris' === 'Bob' // false`
		`5 === 2 + 3 // true`
		`2 === '2' // false; number versus string`

`!==`	Non-equality (is it not the same?)	`5 !== 2 + 4 // true`
		`'Chris' !== 'Bob' // true`
		`5 !== 2 + 3 // false`
		`2 !== '2' // true; number versus string`

Operators

<	Less than	`6 < 10 // true` `20 < 10 // false`
>	Greater than	`6 > 10 // false` `20 > 10  // true`

Conditionals

```
if (userGuess === randomNumber) {
  lastResult.textContent = 'Congratulations! You got it right!';
  lastResult.style.backgroundColor = 'green';
  lowOrHi.textContent = '';
  setGameOver();
} else if (guessCount === 10) {
  lastResult.textContent = '!!!GAME OVER!!!';
  lowOrHi.textContent = '';
  setGameOver();
} else {
  lastResult.textContent = 'Wrong!';
  lastResult.style.backgroundColor = 'red';
  if(userGuess < randomNumber) {
    lowOrHi.textContent = 'Last guess was too low!';
  } else if(userGuess > randomNumber) {
    lowOrHi.textContent = 'Last guess was too high!';
  }
}
```

Events

```
guessSubmit.addEventListener('click', checkGuess);
```

Loops

```
const fruits = ['apples', 'bananas', 'cherries'];
for (const fruit of fruits) {
  console.log(fruit);
}
```

Loops

```
<script>

  let s = 0;
  for(let i=1;i<=10;i++) {
    s += 10;
  }

</script>
```

Debugging JavaScript