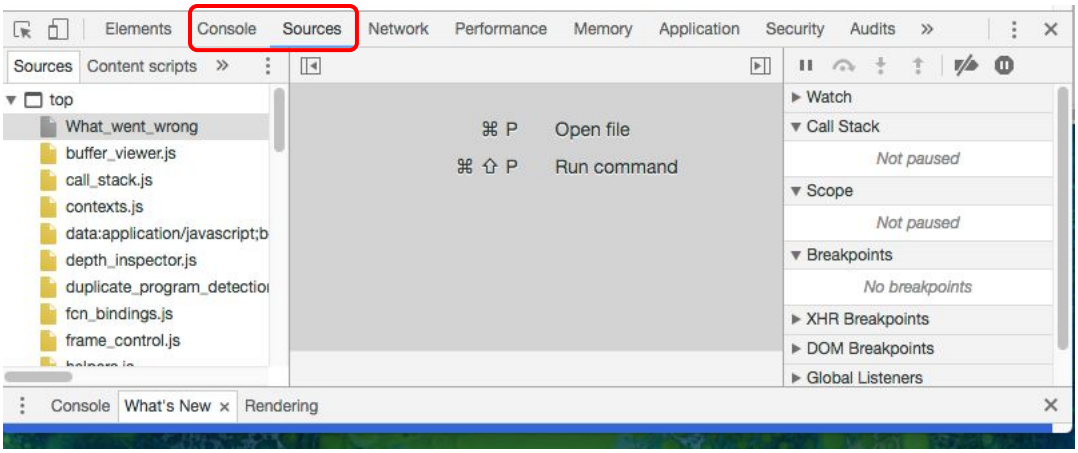
Using TypeScript in Browser

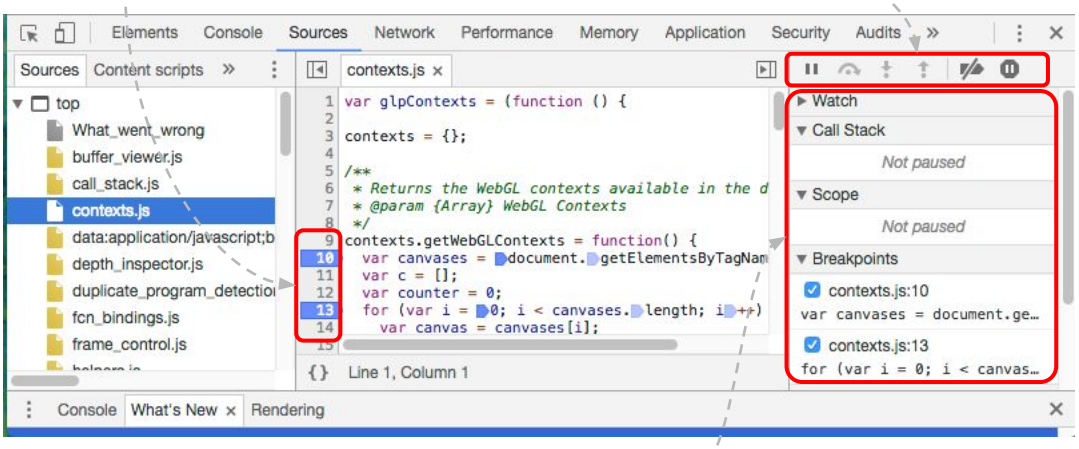
Browser Developer Tool (Chrome)



Browser Debugger (Chrome)

Debugging breakpoints

Debugger Controls (step over, step into, ...)



Variable Inspector

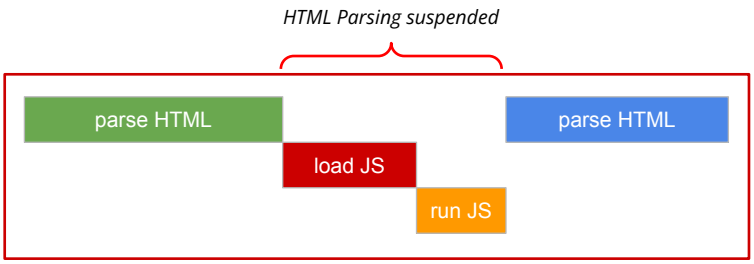
Including JS code in HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="code1.js"></script>
  </head>
  <body>
    <!--
       other HTML contents go here
    -->

    <script src="code2.js"></script>
  </body>
</html>
```

- *Scripts that do not modify page contents are placed in <head>*
- *Scripts that do are placed towards the **end** of <body>*

Script: Loading & Running

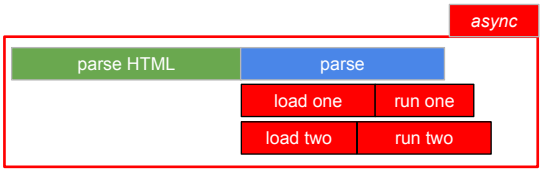
```
<html>
  <head>
    <title>
    <meta ...>
  </head>
  <body>
    <!-- more HTML here -->
    <script src=".....">
    </script>
    <!-- more HTML here -->
  </body>
</html>
```

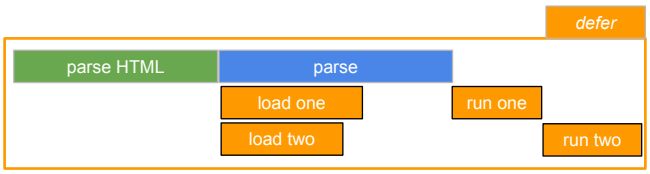


Defer vs Async

```
<html>
  <xxxxx>
    <!-- more HTML here -->
    <script src="one" async></script>
    <script src="two" async></script>
    <!-- more HTML here -->
  </xxxx>
</html>
```

```
<html>
  <xxxxx>
    <!-- more HTML here -->
    <script src="one" defer></script>
    <script src="two" defer></script>
    <!-- more HTML here -->
  </xxxxx>
</html>
```





- *Use* `async` *when you can*
- *Use* `defer` *if you have to*

TS ‹script› option #1: Babel

with babel-standalone

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/babel" src="code1.ts"></script>
  </head>
  <body>
    <!-- other HTML contents go here -->
    <script type="text/babel" src="code2.ts"></script>
  </body>
</html>
```

- *DO NOT use Babel standalone for production*
- *Use transpiled JS for production with bundler (webpack, parcel, rollup, ...)*

TS ‹script› option #2: ParcelJS

```
npm init -y
npm install -save-dev parcel

# Create your-file.html with <script>
# Create one.ts and two.ts

npx parcel serve your-file.html

# Go to localhost:xxxx (in a browser)
```

```
<html>
  <head>
    <script src="one.ts"></script>
  </head>
  <body>
    <!-- more contents here -->
    <script src="two.ts"></script>
  </body>
</html>
```

```
// one.ts
console.debug("Hello from one");
```

```
// two.ts
console.debug("Hello from two");
```

Demo: Parcel Setup & Serve

Browser predefined classes

- Classes associated with individual HTML tags

Tag	Class
`<a>`	`HTMLAnchorElement`
`<body>`	`HTMLBodyElement`
`<button>`	`HTMLButtonElement`
`<img>`	`HTMLImageELement`
`<p>`	`HTMLParagraphElement`

and many more ...

- Other classes: `AudioBuffer`, `Bluetooth`, `ByteString`, `Promise`, `Request`, ...

Browser (Predefined) Objects

Frequently used

- `screen`: the computer screen occupied by the browser
- `document`: the current HTML document that hosts the script
  - Provides functions for manipulating the DOM tree
- `window`: the current window where the HTML doc is rendered

Less frequently used

- `history`: page visit history stack
- `localStorage`: the browser persistent sto
- `location`: the browser input box
- *and many more …*

```
for (const z in window) {
  if (typeof window[z] === "object") {
    console.log(z);
  }
}
```

Try this yourself

Browser window predefined functions

- `alert()`: show an info dialog on the browser
- `addEventListener()`: setup event listeners
- `confirm()`: show a yes/no dialog
- `prompt()`: show an input dialog
- `setInterval()`, `setTimeout()`: start a timer
- `clearInterval()`, `clearTimeout()`: reset existing timer
- …
- *and many more …*

```
for (const z in window) {
  if (typeof window[z] === "function")
    console.log(z);
}
```

Try this yourself

Complete documentations: Web Windows API (MDN: Mozilla Dev Network)

HTML Document CRUD methods/functions

Create	`document.createElement(), document.createTextNode()`
Read	`___.getElementById()` // SINGULAR `___.getElementsByTagName(), ___.getElementsByClassName()` // PLURAL `___.querySelector()` // SINGULAR: search by CSS selectors `___.querySelectorAll()` // PLURAL: search by CSS selectors
Update	`___.appendChild()`
Delete	`___.removeChild()`

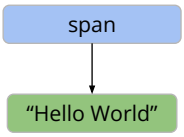
Web Document API

```
for (const z in document) {
  if (typeof document[z] === "function")
    console.log(z);
}
```

Try this yourself

Create Text Nodes

```
<span>Hello world!</span>
```

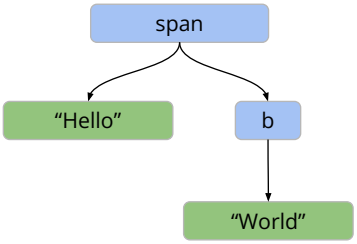



```
// Option 1
const spanParent = document.createElement("span");
const hello = document.createTextNode("Hello World");

spanParent.appendChild(hello);
```

```
// Option 2
const spanParent = document.createElement("span");
spanParent.innerText = "Hello World";
```

Add Multiple Children

```
<span>Hello <b>world!<b></span>
```




```
const spanTop = document.createElement(“span”);

const txt1 = document.createTextNode(“Hello”);
spanTop.appendChild(txt1);

const bChild = document.createElement(“b”);
bChild.innerText = “World”;
spanTop.appendChild(bChild);
```

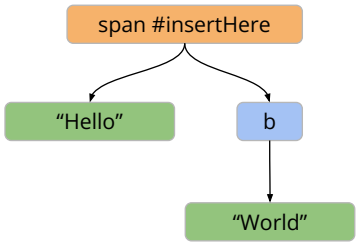

Insert Contents into Existing DOM

before

```
<span id=”insertHere”>
</span>
```

after

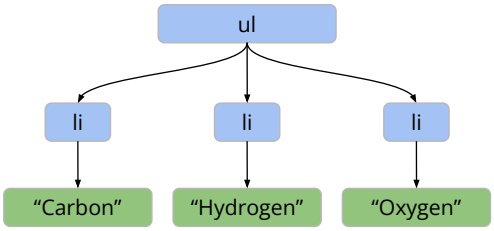
```
<span id=”insertHere”>
  Hello <b>world!<b>
</span>
```




```
const spanTop = document.getElementById(“insertHere”);

const txt1 = document.createTextNode(“Hello”);
spanTop.appendChild(txt1);

const bChild = document.createElement(“b”);
bChild.innerText = “World”;
spanTop.appendCHild(bChild);
```

Add Multiple Children from Array

```
<ul>
  <li>Carbon</li>
  <li>Hydrogen</li>
  <li>Oxygen</li>
</ul>
```




```
const atoms = [“Carbon”, “Hydrogen”, “Oxygen”]
const listTop = document.createElement(“ul”);

for (a of atoms) {
  const atm = document.createElement(“li”);
  atm.innerText = a;
  listTop.appendChild(atm);
}
```


Setting attributes

```
<a
  id=”intro”
  class=”deepIndent noAds”
  href=”http://go.org”>
    Some text here
</a>
```

```
const sample = document.createElement(“a”);

sample.innerText = “Some text here”;
sample.id = “intro”
sample.classList.add(“deepIndent”);
sample.classList.add(“noAds”);
sample.setAttribute(“href”, “http://go.org”);
```

```
const sample = document.createElement(“a”);

sample.innerText = “Some text here”;
sample.setAttribute(“id”, “intro”);
sample.setAttribute(“class”, “deepIndent noAds”);
sample.setAttribute(“href”, “http://go.org”);
```

Live Demo:
(a) VSCode AND (b) Browser Debugger

CSS Selectors & querySelector(All)

querySelector(): select ONE element

```
<body>
  <p class="title">Ice Cream Flavor:</p>
  <ul>
    <li>Death by Chocolate</li>
    <li>Mint Chocolate Chip</li>
    <li>Strawberry</li>
    <li>Bluemoon</li>
  </ul>
  <script src="ice.ts">
</body>
```

Ice Cream Flavors:

- **Too much Chocolate**
- Mint Chocolate Chip
- Strawberry
- BlueMoon

```
const item:Element = document.querySelector("ul > li");
item.textContent = "Too much Chocolate";
```


querySelectorAll(): select MULTIPLE elements

```
<body>
  <p>Ice Cream Flavor:</p>
  <ul>
    <li>Death by Chocolate</li>
    <li>Mint Chocolate Chip</li>
    <li>Strawberry</li>
    <li>Bluemoon</li>
  </ul>
  <script src="ice.ts">
</body>
```

Ice Cream Flavors:

- Death by Chocolate **(on sale)**
- Mint Chocolate Chip **(on sale)**
- Strawberry
- BlueMoon

```
let items:NodeListOf<Element>;
items = document.querySelectorAll("ul > li");
for (let flav of items) {
  if (flav.textContent.includes("Chocolate"))
    flav.textContent = flav.textContent + " (on sale)";
}
```

for-loop is required!

CSS Selector and querySelector(All)

```
<h2>Some heading</h2>
<p>First paragraph</p>
<ol>
  <li class="fruit">Strawberry</li>
  <li class="device">Raspberry Pi</li>
  <li class="singer">Barry Manilow</li>
</ol>

<p>Second paragraph</p>
```

```
const q1 = document.querySelector("h2 + p");
q1.classList.add("red"); // Affect "First paragraph"

const q2 = document.querySelector("h2 ~ ol > li:first-child");
q1.classList.add("red"); // Affect "Strawberry"

const q3 = document.querySelector("li:last-child");
q1.classList.add("red"); // Affect "Barry Manilow"
```

```
const pars = document.querySelectorAll("h2 ~ p");
for (let x of pars) {
  x.setAttribute("___", "___");   // Apply to "First paragraph" and "Second paragraph"
}

const who = document.querySelectorAll("ol > li.singer");
for (let x of who) {
  //  Apply to "Barry Manilow"
}
```


Using Timer

```
<body>
  <p>Ice Cream Flavor:</p>
  <ul>
    <li>Death by Chocolate</li>
    <li>Mint Chocolate Chip</li>
    <li>Strawberry</li>
    <li>Bluemoon</li>
  </ul>
  <script src="ice.ts">
</body>
```

Ice Cream Flavors:

- **Too much Chocolate**
- Mint Chocolate Chip
- Strawberry
- BlueMoon

2 seconds later

```
setTimeout(someFunc, delayInMillisec)
```

```
function choco() {
  const item:Element = document.querySelector("ul > li");
  item.textContent = "Too much Chocolate";
}

setTimeout(choco, 2000);
```

JavaScript Events

Source of Event	Events
Window	onload, onresize, onunload, …
Document	onkeydown, onkeyup, onmousedown, onmouseup, onmouseenter, onmouseleave, …
Input field	onblur, onfocus, onchange, ….
Button	onclick, ondblclick

Complete Reference: Event APIs

Setting Up Event Handlers

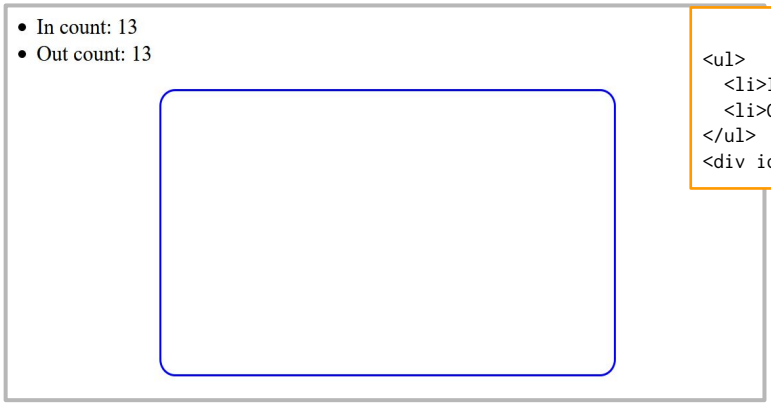
- Which Event?
- Who is the event recipient or the event source?
  - Resize => window
  - Key presses => document
  - Load => document
  - Click => button, image, ….
  - Focus => input elements
  - Mouse => elements
- Details of the event object properties (MouseEvent, KeyboardEvent, ….). Refer to online API

```
function keyHandler(ev: KeyboardEvent): void {
  // put code here
}

function clickHandler(ev:MouseEvent): void {
  // put code here
}

document.addEventListener("keypress", keyHandler);

const myLogo = document.getElementById("____");
myLogo.addEventListener("click", clickHandler);
```

CodePen: Event Handling Demo

Counting Mouse Traffic

Screenshot (from CodePen.io)



HTML

```
<ul>
  <li>In count: <span id="one"></span></li>
  <li>Out count: <span id="two"></span></li>
</ul>
<div id="mybox"></div>
```

CSS

```
#mybox {
  border: 3px solid blue;
  min-height: 400px;
  max-width: 60%;
  margin: auto;
  border-radius: 24px;
}
```

Counting Mouse Traffic

HTML

```
<ul>
  <li>In count: <span id="one"></span></li>
</ul>
<div id="mybox">

</div>
```

```
const a:HTMLSpanElement = document.getElementById("one");
const box:HTMLElement = document.getElementById("mybox");
let enterCount = 0;
a.innerText = enterCount.toString();

function enterBox(): void {
  enterCount++;
  a.innerText = enterCount.toString();
}


box.addEventListener("mouseenter", enterBox);
```

TS