Beginnen met functies, arrays en objecten

Overzicht

- Wat zijn functies
- Hoe definieert en gebruikt u functies?
- Op welke manier worden parameters meegegeven bij een functieaanroep?
- Hoe functies waarden kunnen retourneren
- Hoe arrays zijn opgebouwd
- Welke functies voor arrays beschikbaar zijn
- Hoe objecten in JavaScript worden gemaakt en gebruikt

Functies – Herhaald taken uitvoeren

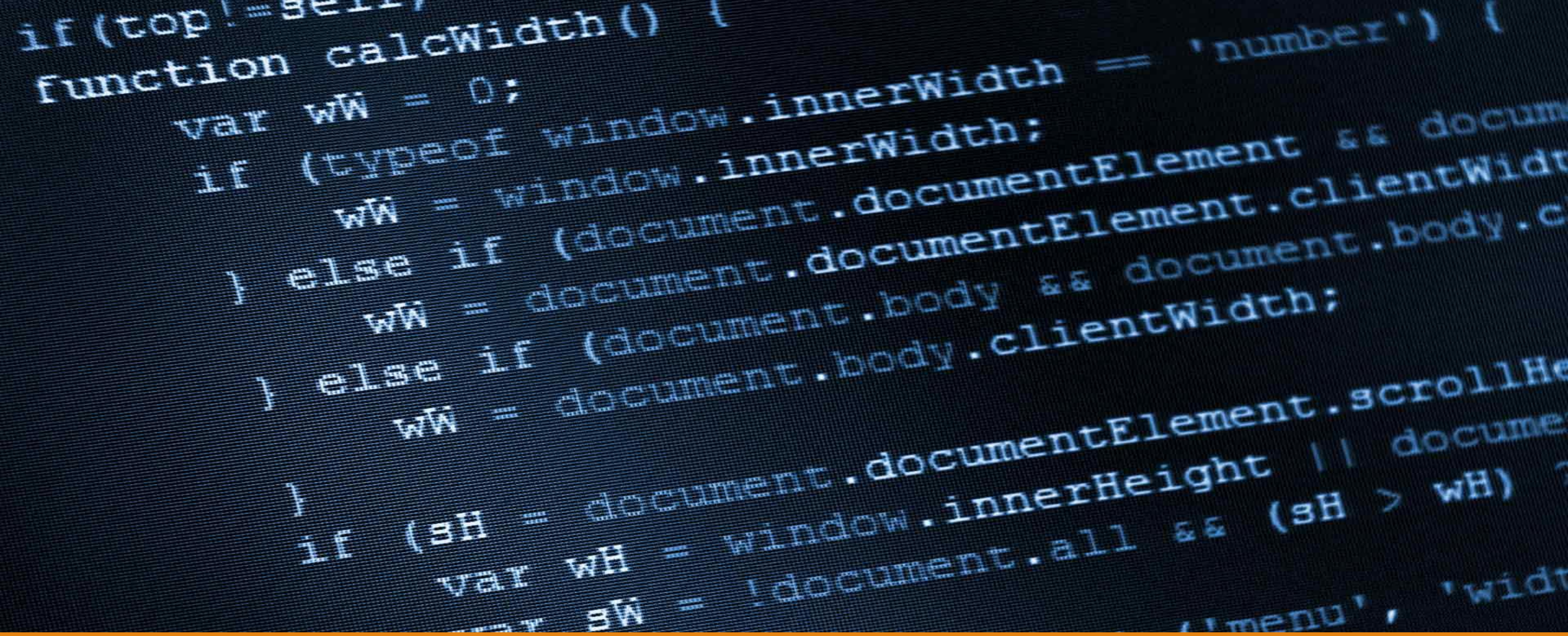
- Als in een script een taak meer dan eens voorkomt is het nuttig voor die taak een functie te schrijven
- Een functie is in JavaScript niets meer dan een speciaal soort variabele
- Wordt pas uitgevoerd als deze aangeroepen wordt
- Eerst wordt een functie gedefinieerd en vervolgens aangeroepen

Functies – Parameters

- Is een waarde die door de functie intern wordt gebruikt
- Staat tussen haakjes bij de functieaanroep en –definitie
- Zijn niet verplicht
  - Zelfs als een functie met vier parameters wordt gedefinieerd, dan mag in het script de functie aangeroepen worden met minder parameters

Structuur van een functie

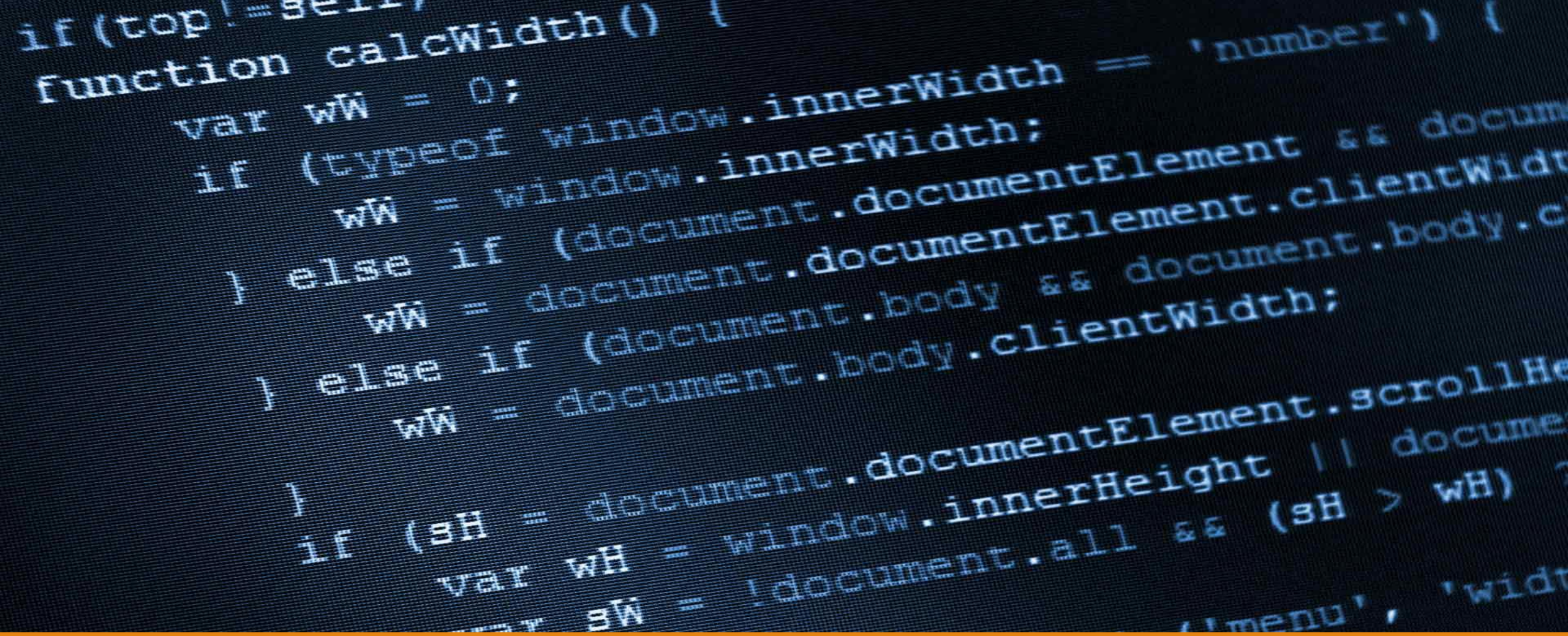
- **function:** Eerst komt het keyword “function”
- **functienaam:** Naam van de functie. Zelfde regels als variabelen
- **parameters:** Achter de naam staat tussen haakjes een lijst met parameters. Het haakjespaar blijft leeg als de functie geen parameters nodig heeft. De haakjes zelf zijn altijd verplicht. De parameters worden met komma’s gescheiden
- **Body:** De body van de functie staat tussen accolades {…}


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Andere notatie

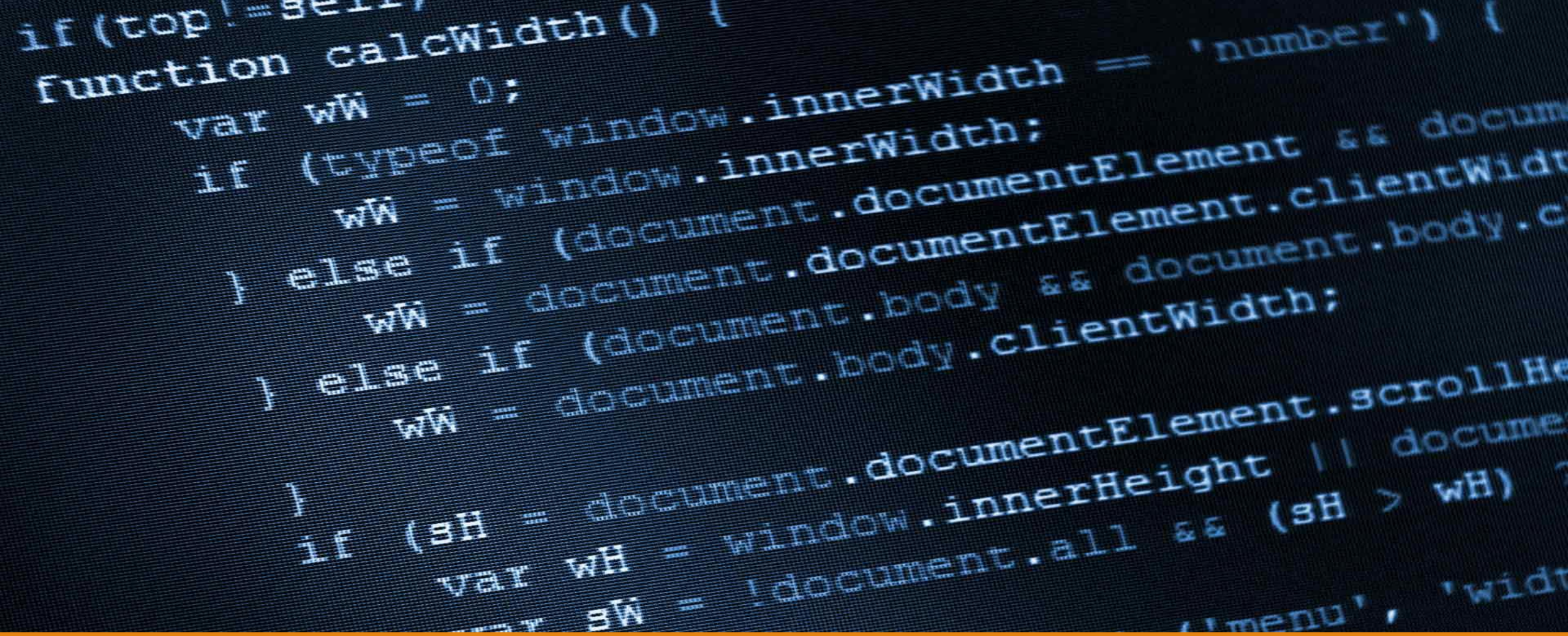
- Omdat een functie in JavaScript gewoon een variabele is, mag je ze ook definiëren gelijk een variabele


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Anonieme functies

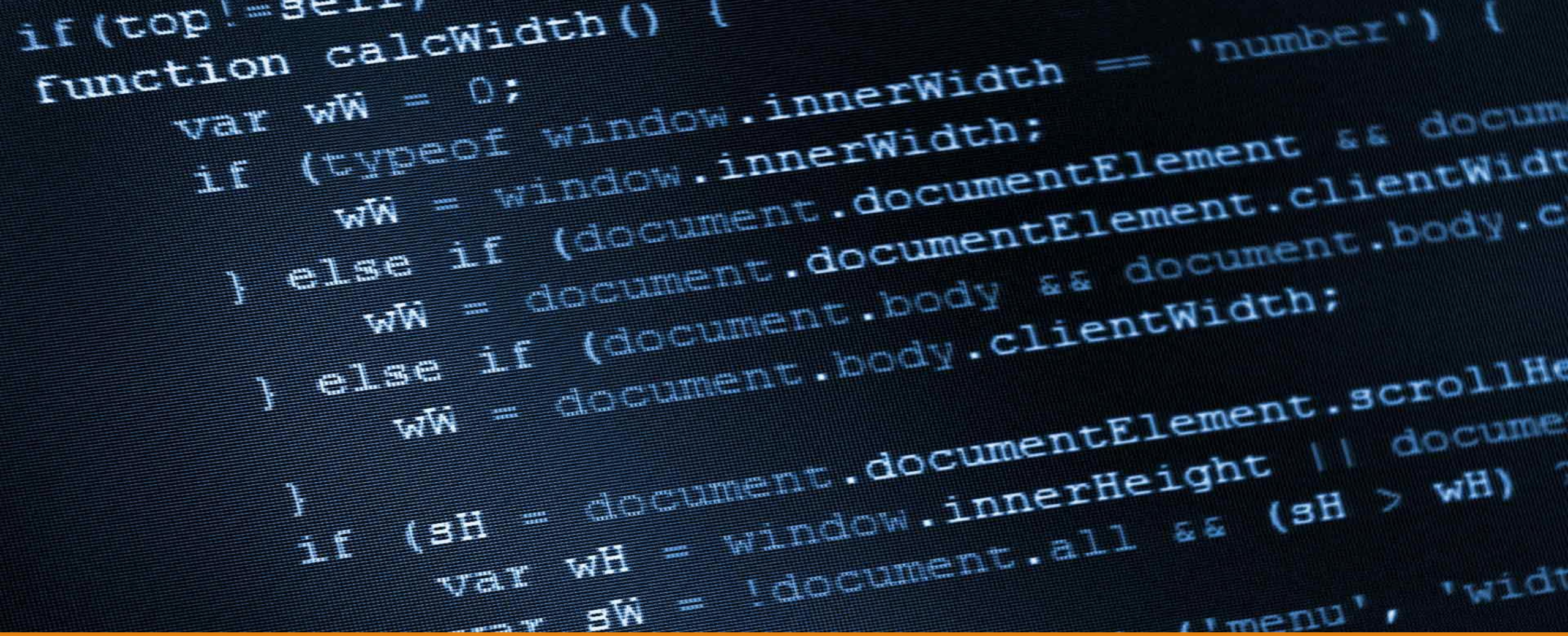
- Een functie die geen naam heeft
- Komen regelmatig voor
- Kan niet later vanuit het script worden aangeroepen
- Word ter plekke aangemaakt en uitgevoerd, daarna terug verwijderd uit het geheugen


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Parameters doorgeven

- Parameters binnen en buiten de functie hoeven niet dezelfde naam te hebben
- Inhoud wordt doorgegeven aan de parameters van de functie


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Regels voor parameters

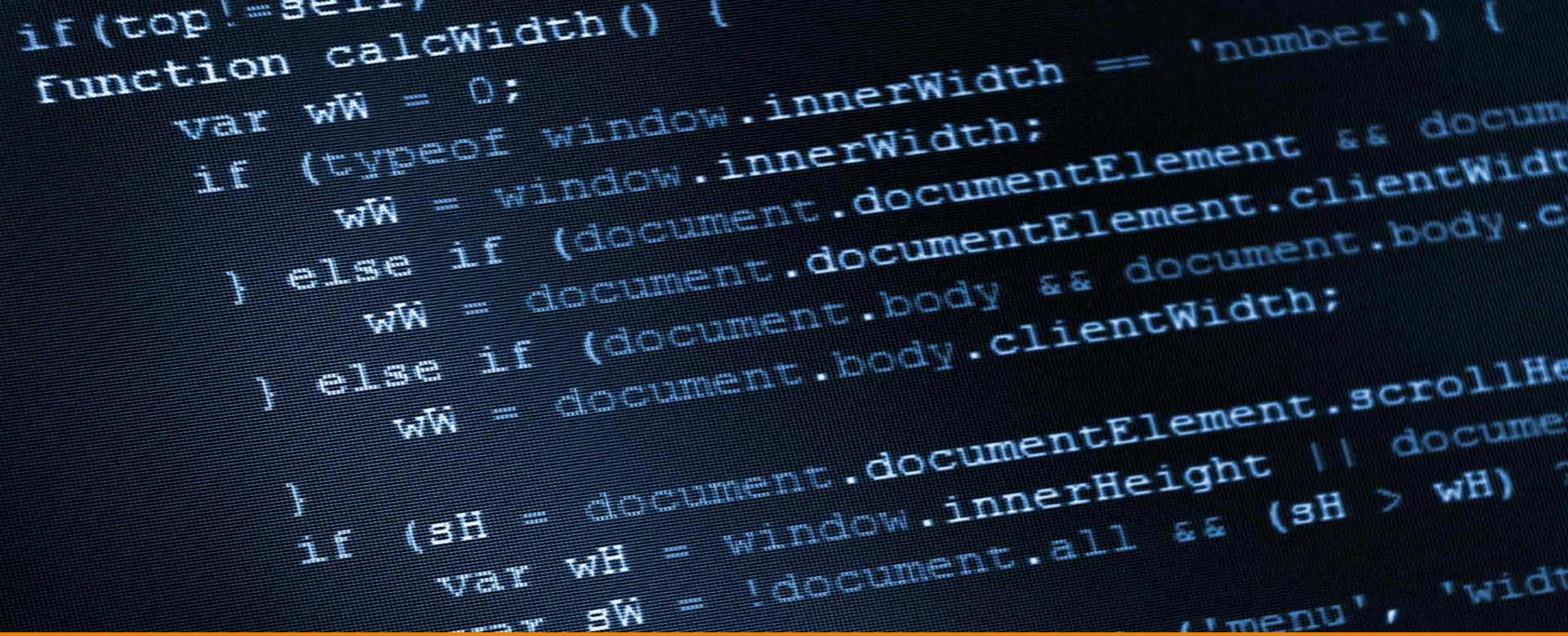
- Alle typen variabelen mogen worden gebruikt als parameters voor functies. Zelfs een andere functie meegeven als parameter is toegestaan
- Wijzigingen van de variabele binnen de functie hebben geen invloed op de waarde van de originele variabele (buiten de functie)
- De parameters bestaan alleen zolang de functie wordt uitgevoerd. Bij het opnieuw uitvoeren van de functie worden de waarden van de variabele telkens ‘ververst’

Waarden retourneren

- Hiervoor gebruiken we het statement **“return”**
- De teruggegeven waarde kan worden opgeslagen in een variabele om hem later in het script te gebruiken

Één waarde retourneren

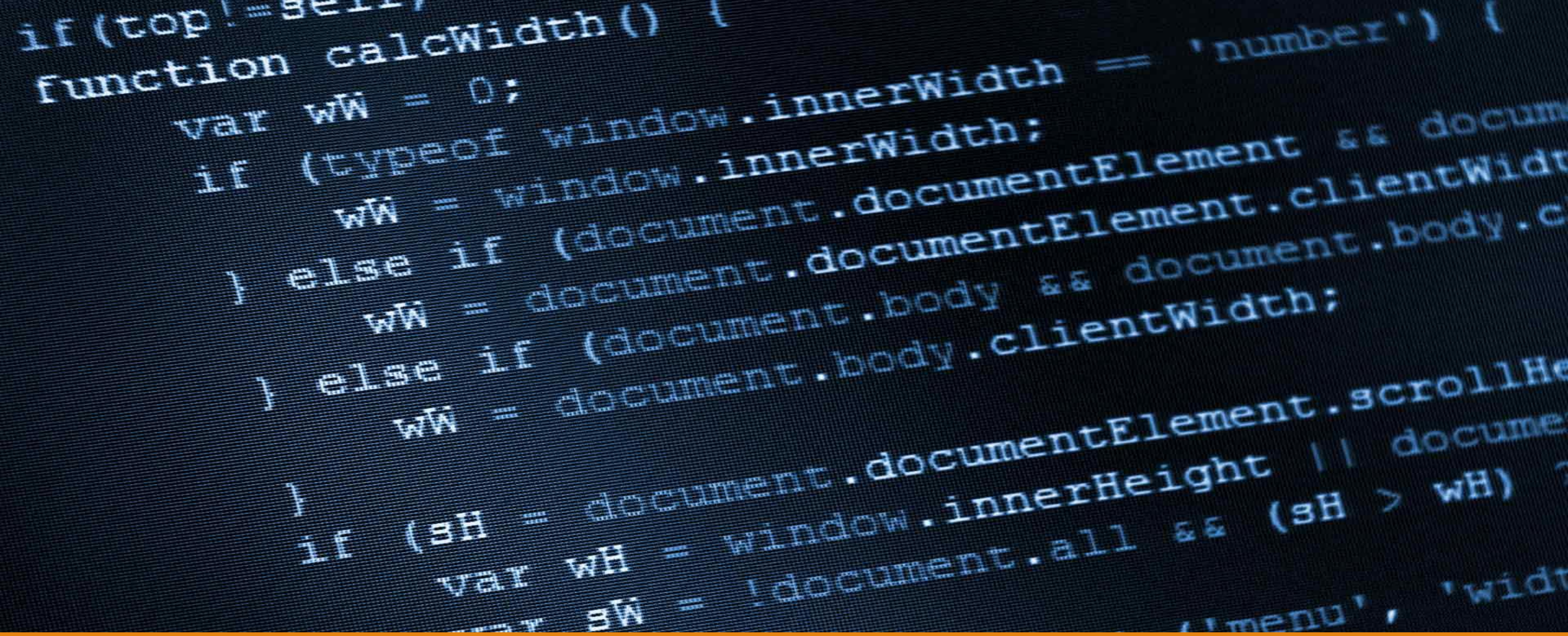
- Per functie kan maar één waarde worden teruggegeven.
- Bij meerdere returns zal alleen de eerste waarde worden teruggegeven
- Na het statement “return” wordt de functie beëindigd.
  - het script zal verder gaan met uitvoeren van het eerstvolgende statement na de plek waar de functie oorspronkelijk werd aangeroepen


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Meerdere waarden retourneren

- Wilt u een functie in JavaScript meerdere waarden laten retourneren, dan moet deze altijd ingekapseld *(wrappen)* worden in een object


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

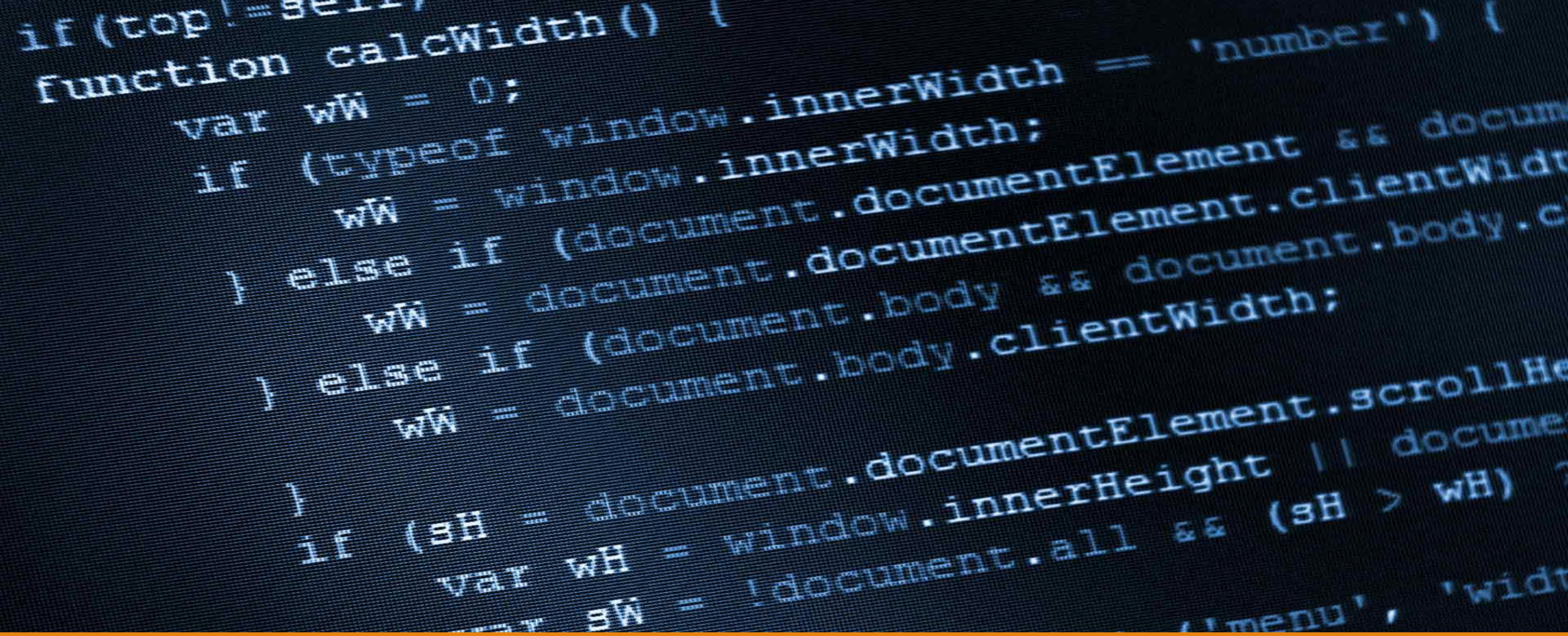
Demo

Arrays

- Een geordende verzameling elementen
- Elk element heeft een genummerde positie in de array. Dit wordt de index genoemd
- Arrays beginnen altijd te tellen bij nul (0)
- Een array wordt in JavaScript genoteerd met blokhaken []
- Arrays zijn dynamisch
  - Het is niet nodig om van tevoren aan te geven hoe groot de array moet worden
  - Elementen mogen uit verschillende gegevenstypen bestaan
  - Elke array heeft de eigenschap ‘length’. Hiermee is het aantal elementen in de array op te vragen

Arrayelementen uitlezen en toevoegen

- De inhoud van een array kan opgevraagd worden met de notatie [x]
  - Op de plaats van de ‘x’ komt de index te staan


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

.join()

- Converteert alle elementen in de array naar een string en voegt deze samen tot één lange string
- Scheidingsteken kan meegegeven worden
  - Standaard een komma (,)
- Omgekeerd: **.split()**

.reverse()

- Keert de volgorde van de elementen in de array om
- Er wordt **geen** nieuwe array aangemaakt!
- Volgorde wordt aangepast in de bestaande array

.sort()

- Sorteerd de elementen binnen een array
- Standaard wordt er op alfabet gesorteerd
- Indien nodig worden de elementen eerst omgezet naar strings
- Er kan een eigen sorteer functie meegegeven worden voor complexe objecten te sorteren

.push() & pop()

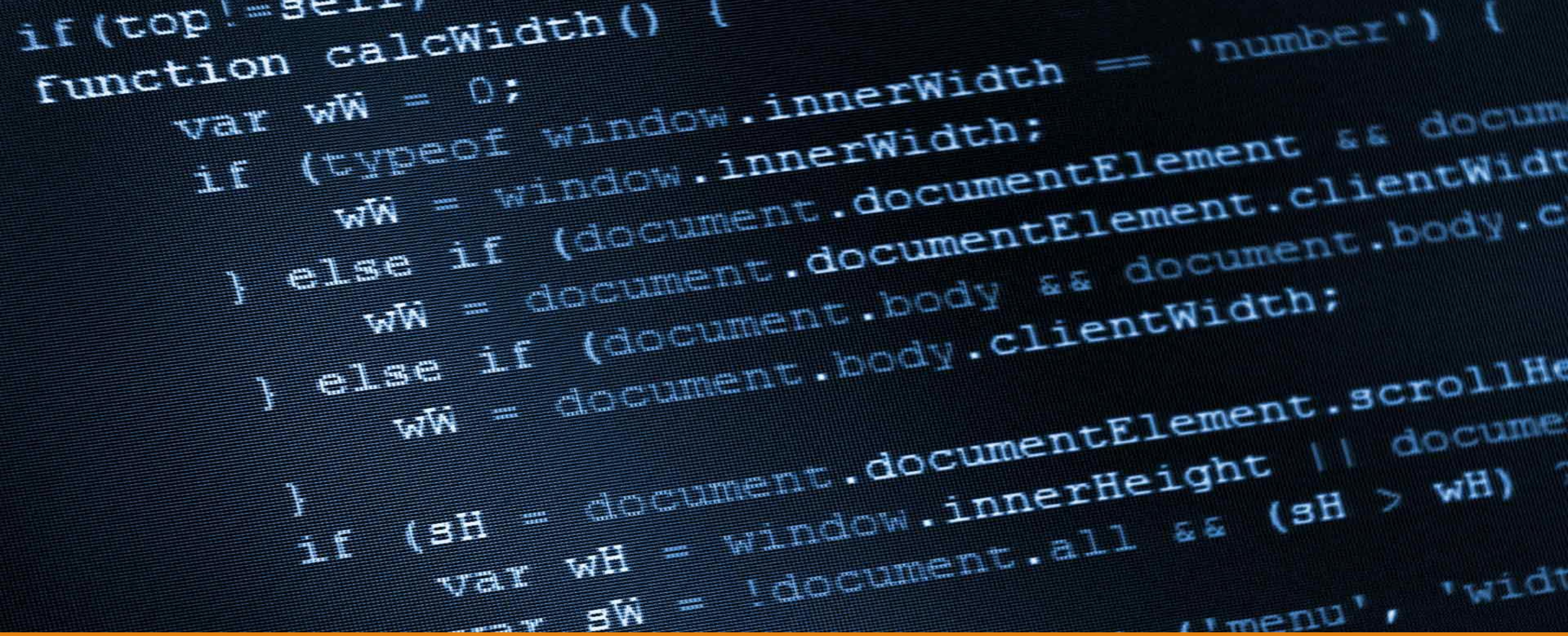
- **push:** Element aan het einde van de array toevoegen
- **pop:** Verwijdert het laatste element van de array

Overige arraymethoden

- **.concat():** Voegt twee arrays samen tot een nieuwe array
- **.slice():** Retourneert een subarray uit de aangegeven array
- **.splice():** Voegt elementen in of verwijdert elementen uit de array op de aangegeven positie
- **.shift():** Zoals .push(), maar voegt element toe aan het begin van de array

Overige arraymethoden

- **.unshift():** zoals .pop(), maar verwijdert element aan het begin van de array
- **.toString():** Zet alle elementen in de array om naar een string en retourneert deze
- **.toLocaleString():** Idem, maar houdt rekening met het scheidingsteken dat voor de huidige taal versie gebruikt wordt


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

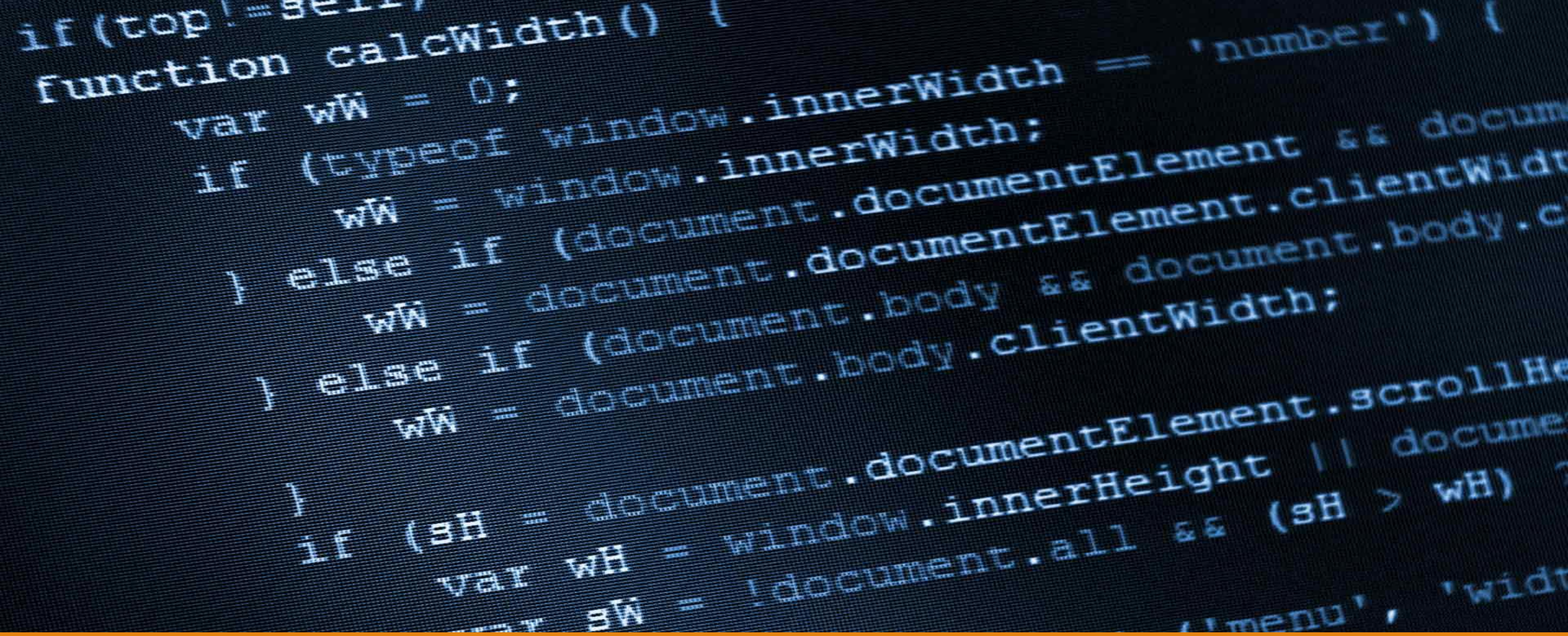
Demo

Objecten
Eigenschappen, namen en waarden

- Gedefinieerd als een container van eigenschappen *(properties)*
- Elke eigenschap heeft een **naam** en een **waarde**
- De naam en waarde worden met een dubbele punt van elkaar gescheiden
- De naam van een eigenschap is een string, de waarde van een eigenschap is elke geldige JavaScript-waarde
- De eigenschappen binnen het object worden met komma's van elkaar gescheiden

Classes

- Generieke objecten met telkens andere waarden voor titel, auteur, etc…
- Javascript manier om classes te definiëren


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Program flow controleren

Overzicht

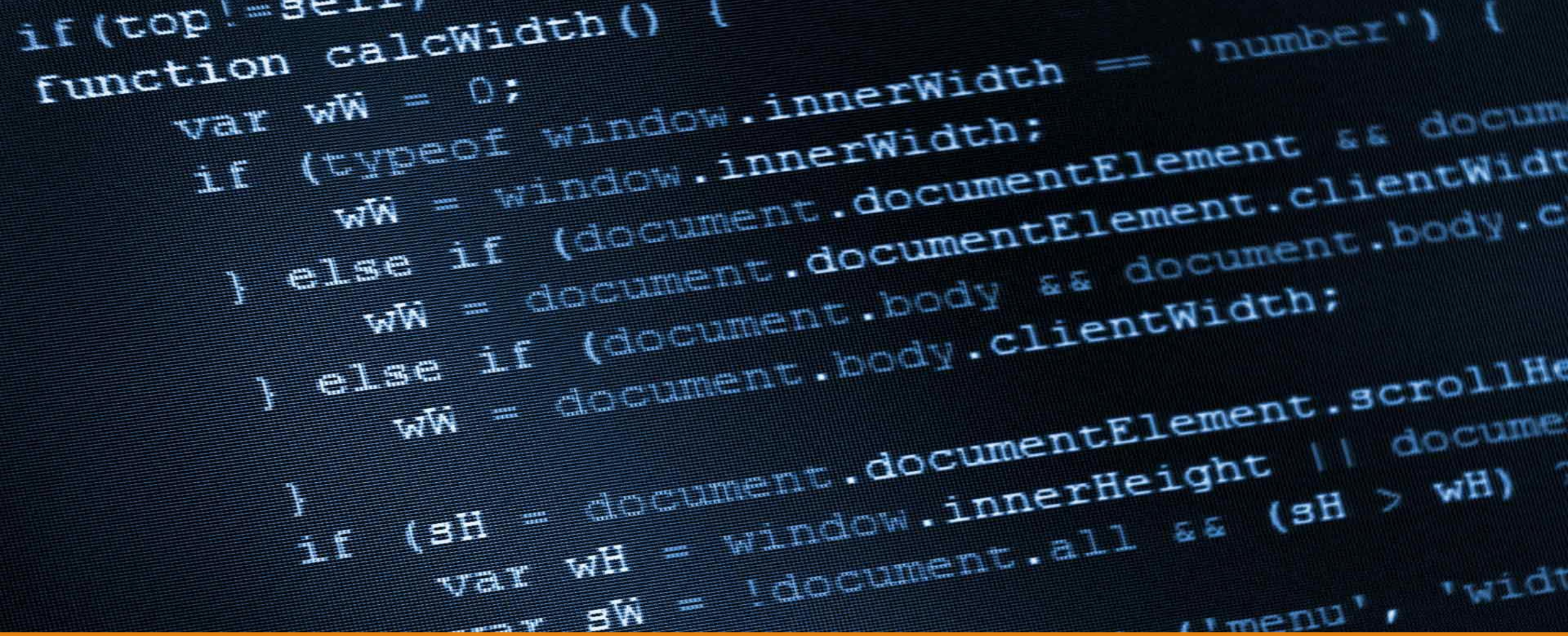
- Conditionele statements: **if-else** en **switch()**.
- Hoe lussen **while()** en **for()** worden gebruikt
- Het object gebonden statement **for-in**
- Hoe deze statements in echte programma’s worden ingezet

Verschillende typen lussen

- **Voorwaardelijke of conditionele statements:** Dit zijn de statements if() en switch(). Ze zorgen ervoor dat JavaScript de ene of juist de andere opdracht uitvoert, afhankelijk van de waarde van een expressie
- **Lussen of loops:** Dit zijn de statements while() en for(). Ze voeren een opdracht een bepaald aantal keren achter elkaar uit, vaak ook weer op basis van de waarde van een expressie
- **Jumps:** Dit zijn de statements break, continue, return en throw. Ze onderbreken de uitvoering van een lus of vervolgen deze juist, ook weer op basis van de uitkomst van een expressie

If-else

- Controleren of er voldaan is aan een bepaalde expressie
  - **true**: statements na de **if** worden uitgevoerd
  - **false**: statements na de **else** worden uitgevoerd
- Kunnen meerdere **else** blokken voorkomen


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

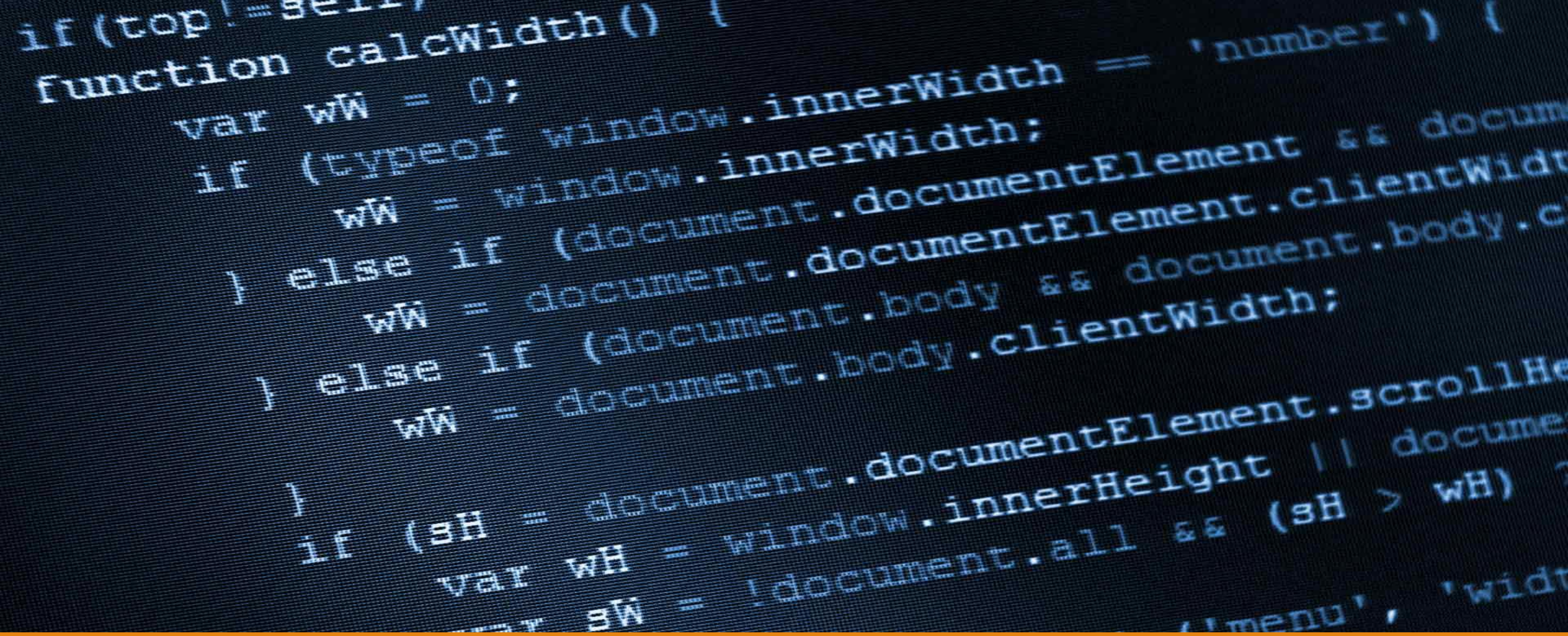
Demo

While()

- Opdrachten binnen het statement worden uitgevoerd zolang aan een bepaalde voorwaarde wordt voldaan

for()

- Altijd 3 parameters
  - De eerste parameter *(index)* specificeert een variabele en kent hieraan een waarde toe. Hier wordt de begintoestand van de lus bepaald
  - De tweede parameter *(voorwaarde)* is een voorwaarde die naar true moet evalueren om de lus te laten herhalen. Zodra de voorwaarde naar false evalueert, wordt de lus beëindigd. Dit deel van de for-lus heet dan ook letterlijk de voorwaarde
  - De derde en laatste parameter *(indexbewerking)* is een statement dat bij elke herhaling van de lus wordt uitgevoerd. Meestal wordt hier de index verhoogd *(of verlaagd)*. Het statement ziet er vaak uit als **index += 1** of **index -= 1**


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

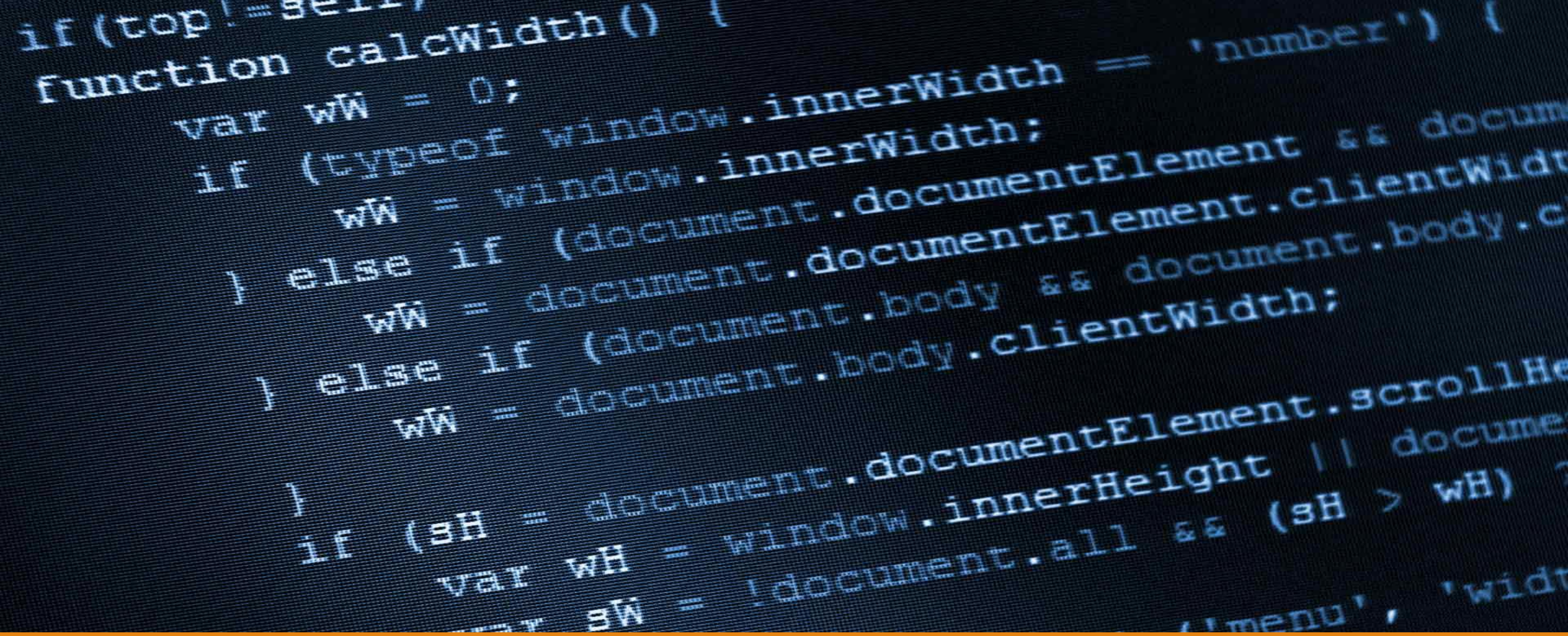
Demo

De statements break, continue en return

- **break:** Met break wordt een lus of ander statement expliciet beëindigd. Na break; gaat de uitvoering van het script door vanaf het eerste statement dat volgt op de sluitaccolade. Het is alleen geldig binnen lus structuren
- **continue:** Dit slaat de rest van de lus over als aan een bepaalde voorwaarde wordt voldaan. Maar in tegenstelling tot break gaat continue verder vanaf de volgende lus doorloop. De lus wordt dus niet afgebroken. Alleen alles voorbij continue wordt eenmalig genegeerd. Continue wordt dan ook vaak gebruikt in combinatie met if.

De statements break, continue en return

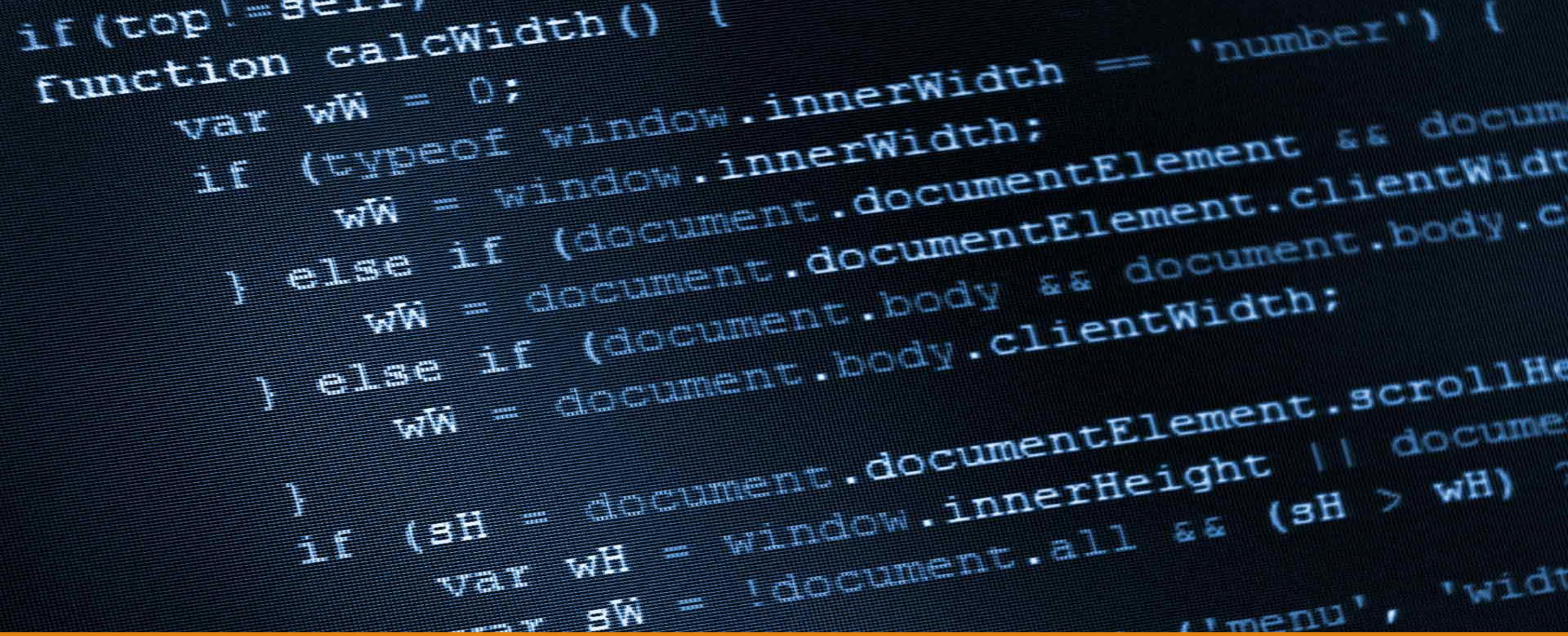
- **Return:** Het statement return; onderbreekt ook de uitvoering van een blok programmacode, maar heeft daarnaast een aantal extra kenmerken
  - return is alleen geldig binnen een functie. Het beëindigt de uitvoering van de functie. JavaScript keert terug naar het aanroepende statement
  - return kan eventueel een waarde teruggeven. Verplicht is dit niet *(break en continue kunnen dit niet)*


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo

Het statement for-in

- Loopen over alle eigenschappen van een object
- De waarden van de eigenschappen van een object worden binnen de lus als een array benaderd


if (top != self)
function calcWidth() {
var wW = 0;
if (typeof window.innerWidth == 'number') {
wW = window.innerWidth;
} else if (document.documentElement && docum
wW = document.documentElement.clientWidt
} else if (document.body && document.body.c
wW = document.body.clientWidth;
}
if (sH = document.documentElement.scrollHe
var wH = window.innerHeight || docume
var sW = !document.all && (sH > wH)

Demo